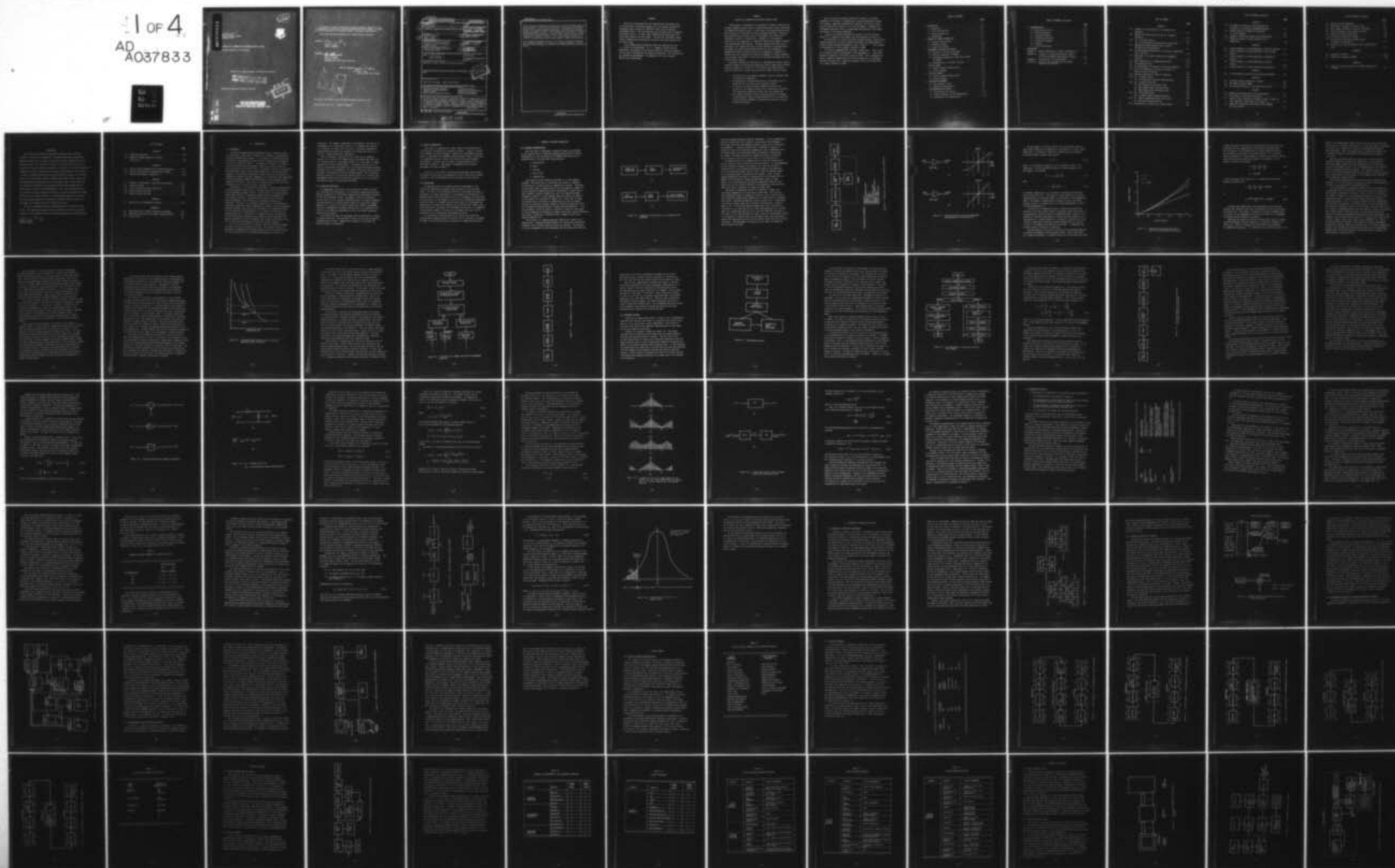
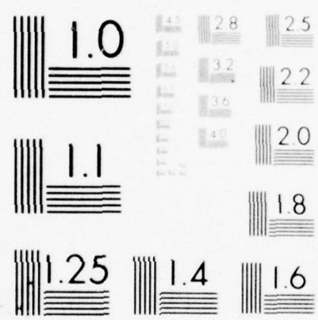
AD-A037 833 GEORGIA INST OF TECH ATLANTA ELECTRONICS TECHNOLOGY LAB F/G 9/2
INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY.(U)
FEB 77 R W MOSS, R W RICE, D R SENTZ F30602-75-C-0247
UNCLASSIFIED RADC-TR-77-42 NL

1 OF 4
AD
A037833





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA037833

INTERACTIVE COMMUNICATION SYSTEMS MODELING

RADC-TR-77-42
Final Technical Report
February 1977

INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY

Georgia Institute of Technology

Laboratory Directors' Fund No. 01717504

ROME AIR DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
GRIFFISS AIR FORCE BASE, NEW YORK 13441

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be available to the general public, including foreign nations.

This report has been reviewed and is approved for publication.

APPROVED:

PETER K. LEONG
Project Engineer

APPROVED:

FRED I. DIAMOND
Technical Director
Communications and Control Division

FOR THE COMMANDER:

JOHN P. HUSS
Acting Chief, Plans Office

This effort was funded totally by the Laboratory Directors' Fund.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

18 19

1. REPORT NUMBER: RADC-TR-77-42

2. GOVT ACCESSION NO.:

3. RECIPIENT'S CATALOG NUMBER: 9

4. TITLE (and Subtitle): 6 INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY.

5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report. June 1975 - August 1976,

6. PERFORMING ORG. REPORT NUMBER: N/A

7. AUTHOR(s): 10
Richard W. Moss
Robert W. Rice
Donald R. Sentz

8. CONTRACT OR GRANT NUMBER(s): 15 F30602-75-C-0247

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Electronics Technology Laboratory
Georgia Institute of Technology
Atlanta GA 30332

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61101F 16 01717504 17 75

11. CONTROLLING OFFICE NAME AND ADDRESS:
Rome Air Development Center (DCLF)
Griffiss AFB NY 13441

12. REPORT DATE: 11 February 1977

13. NUMBER OF PAGES: 320

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same 12 311p.

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report):
Approved for public release; distribution unlimited.

DDC RECEIVED APR 6 1977 C

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report):
Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Peter K. Leong (DCLF)

This effort was funded totally by the Laboratory Directors' Fund.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Interactive Computer Simulation	Time Domain Simulation
Communication Systems Modeling	Modified Monte Carlo Technique
Graphics Display	Man-Machine Interaction
Hierarchical Programming Structure	Modular Software Models

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report deals with an effort to develop an interactive computer model for analysis and design of digital communications systems. In this report concepts are formulated and described for a Model for Interactive Design and Analysis of Communication Systems (MIDACS). Methods of approach for concept formulation are described, examples of previous modeling approaches are presented, characteristics of the MIDACS concept are documented, and a representative computer model is described. (see reverse)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409743

4B

Various interactive approaches are shown along with general hardware and software requirements. The formulated interactive concept includes the computer, display devices, and other input/output devices. Mathematical modeling approaches are described including standard Monte Carlo techniques, deterministic methods, and analytical models. The recommended modeling structure includes a modified Monte Carlo technique to minimize computer time for certain communication system performance measures. A multi-level analysis structure is recommended which includes a hierarchy for the physical model combined with a hierarchy of analysis levels which permits greatest computer use flexibility.

The recommended programming structure is described including the development chain, software composition, and approach to software development. Function and system library elements are identified and MIDACS hardware requirements are stated.

PREFACE

This report was prepared by the Communications Technology Group of the Electronics Technology Laboratory at the Georgia Institute of Technology under Contract No. F30602-75-C-0247 with the Rome Air Development Center. The work described was performed under the general supervision of Mr. R. W. Moss, Head, Communications Technology Group and Project Director for this effort. For Rome Air Development Center, Digital Communications Experimental Facility, the Project Monitor was Mr. Peter Leong.

In the Communications Technology Group the contribution of Mr. J. L. Grover is acknowledged. Dr. J. L. Hammond, Georgia Tech EE Department, participated in the earlier phases of this work primarily in the areas of mathematical and modeling approaches. His contributions are gratefully acknowledged. The guidance of Mr. Peter Leong of RADC is also acknowledged.

SUMMARY

INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY

Traditionally, in assessing the performance of digital communications systems, designers usually resort to highly specialized studies addressing the immediate problems at hand. In many cases, the design approach places heavy reliance on "textbooks" and other published materials. Frequently much previous work, because of lack of knowledge or the difficulty in information retrieval, is duplicated. Moreover, it is frequently impractical if not impossible to completely characterize digital communication systems using this approach. Although computers have been used to advantage, their capabilities have not been exploited to maximum benefit. Put simply, the analysis and design approach for the technology of communications has not kept pace with the technology of computers and displays and their capability for general assessment of digital communications systems, particularly in the area of high speed processing and interactive graphic displays.

This study examined the possibility of implementing a highly flexible computer program for modeling and simulating digital communication systems. The effort began with a review of the literature describing previous efforts in this area. Several conclusions were drawn from this survey:

1. Virtually all of the analysis programs currently available lack the interactive capability,
2. A high degree of program interactivity is desirable since it allows the user to carry out his analysis quickly,
3. An effective trade-off between computation time and the accuracy of the computed results can be realized by incorporating a multilevel modeling structure into the program, and
4. The commonly used Monte Carlo simulation approach is probably the most general analysis scheme available, but it is also one of the most time consuming, and therefore alternate procedures should be developed.

Each of the above conclusions served as a basis for continued examination of the means of implementing a general modeling and simulation program. Modeling techniques were explored, and the more useful ones are described in Section 2.2 of the Final Report. Basically the recommended approaches result in time-domain descriptions of the inputs and outputs of the system components.

A variety of simulation methods were studied, and the ones most useful for a program of the present type are discussed in Section 2.3.

The practicality of each of the major program concepts, interactivity, multilevel analysis, and block diagram structuring, was demonstrated in a small software program developed as a part of this study. This program was developed for a GT-44 graphics terminal at DICEF.

It has been recommended that the basic concepts developed as a part of this study be used as guidelines in the development of a large scale simulation program for the DICEF facility at RADC. Such a program would employ a graphics terminal such as the GT-44 serving as an intelligent satellite processor for a large host computer such as the C-8500 which is currently available at RADC.

TABLE OF CONTENTS

	Page
1. INTRODUCTION	1-1
1.1 Background	1-1
1.2 Scope and Objectives	1-2
1.3 Report Organization	1-3
1.4 Methodology	1-3
2. SUMMARY OF CONCEPT FORMULATION	2-1
2.1 General Considerations	2-1
2.2 Modeling Concepts	2-16
2.3 Simulation Concepts	2-34
3. PROGRAMMING STRUCTURE FOR MIDACS	3-1
3.1 Approach to Software Development	3-1
3.2 Software Composition of MIDACS	3-4
3.3 Programming Languages and Operation Systems for MIDACS	3-6
3.4 Implementation of Software Structure	3-8
4. SYSTEM LIBRARY	4-1
4.1 General Element Characteristics	4-1
4.2 Library Elements	4-3
5. FUNCTION LIBRARY	5-1
5.1 General Element Characteristics	5-1
5.2 Library Elements	5-1
6. HARDWARE REQUIREMENTS	6-1
6.1 Major Hardware Items	6-1
6.2 Hardware Characteristics	6-1
7. THE DEMONSTRATION PROGRAM	7-1
7.1 Illustrating the Concepts	7-1
7.2 Software Approach for the Demonstration	7-9
7.3 Evaluation of the Demonstration	7-16

TABLE OF CONTENTS (continued)

		Page
8.	RECOMMENDATIONS	8-1
	8.1 Applications Survey	8-1
	8.2 Modeling Approach	8-2
	8.3 Simulation Approach	8-2
	8.4 Programming Structure	8-3
	8.5 Software Development	8-4
	8.6 Hardware	8-5
	8.7 Further Development	8-6
BIBLIOGRAPHY		1
APPENDIX A:	MIDACS Demonstration Software Documentation	A-1
APPENDIX B:	Programs for Generating Permanent Display Files Used By MIDACS Demonstration Program	B-1
APPENDIX C:	Documentation for Program MIDSYS	C-1
APPENDIX D:	Subroutine Calling Hierarchy for MIDACS Demonstration Software	D-1

LIST OF FIGURES

Page

SECTION 2

2-1 Examples of the Diversity of the Systems to be Simulated 2-2

2-2 Example of Problem Setup Process for a Computer Simulation 2-4

2-3 (a) An Actual System 2-5

(b) The Simulation Representation of That System 2-5

2-4 Mathematical Approximations Used to Model a Voltage-Current Relationship 2-7

2-5 The Relationship Between Level of Analysis, Computation Time, and Error 2-12

2-6 Flow Chart for a Highly Interactive Programming Structure 2-14

2-7 Basic Components of a Communication System 2-15

2-8 The Modeling Process 2-17

2-9 The Development of System and Function Level Models . . . 2-19

2-10 The Block Diagram for a System Level Model of a Binary Coherent PSK System 2-21

2-11 Functions Described by Algebraic Equations 2-25

2-12 (a) A Lowpass Filter 2-26

(b) Its Differential Equation Representation 2-26

2-13 (a) A Signal's Spectrum 2-30

(b) When Sampled at the Nyquist Rate 2-30

(c) When Sampled Below the Nyquist Rate 2-30

(d) When Sampled Above the Nyquist Rate 2-30

2-14 (a) A Functional Block 2-31

(b) Its Model Using a Data Reconstructor 2-31

2-15 (a) Typical Communication System Structure 2-42

(b) Equivalent System Structure 2-42

2-16 Illustration of an Error for a Binary System 2-44

LIST OF FIGURES (continued)

Page

SECTION 3

3-1 An Example of Subroutine Class Groupings and Hierarchy for Structured Programming Approach 3-3

3-2 Example Showing the Numerical Description of a Block Diagram 3-5

3-3 Diagram of Simulation Process Showing Data Paths 3-7

3-4 MIDACS Environment in Which Software will be Implemented 3-10

SECTION 4

4-1 A Block Diagram of a System Employing a Satellite Repeater. 4-5

4-2 A Block Diagram of a System Employing a Line-of-Sight Channel 4-6

4-3 A Block Diagram of a System Employing a Troposcatter Channel 4-7

4-4 A Block Diagram of a System Employing a Wire-Line Channel 4-8

4-5 A Block Diagram of a System Employing an HF Channel 4-9

SECTION 5

5-1 A Block Diagram Constructed With Functional Elements . . . 5-2

SECTION 6

6-1 The Major Hardware Components of MIDACS 6-2

6-2 The C8500 Interface Scheme 6-3

6-3 The Interface Details for the C8500 and the GT-44 6-4

SECTION 7

7-1 Example Run Showing Queries and User Responses 7-2

7-2 User-Computer Interactions and Results 7-3

7-3 Sample of Text Information Available to User As Result of 'Yes' Response to Last Question in Figure 7-1 7-5

7-4 Sample Run Illustrating Some System Level Commands and Their Results 7-6

LIST OF FIGURES (continued)

		Page
7-5	Editing Function Parameters	7-7
7-6	Sample Error Messages at Function Level	7-7
7-7	Root Segment and Region 1 Overlays 1 Through 4	7-10
7-8	Function Elements in the Overlay Structure	7-11
7-9	Function Configuration Routines	7-12
7-10	Old Version System Level Package	7-13
7-11	New Version System Level Package as Used in MIDSYS (See Appendix C)	7-14
7-12	Structure for Text and Graphics I/O For Inexperienced Users	7-15

APPENDIX A

A-1	Operation of PRCTRL	A-81
A-2	Operation of Subroutine SMPHLD	A-108

APPENDIX D

D-1	Subroutine Calling Hierarchy for MIDACS Demonstration Software.	D-2

EVALUATION

This effort was established under RADC Laboratory Director's Fund Program to initiate the development of a general interactive computerized simulation model for the analysis and design of digital communication transmission systems. The tasks were to survey the existing state-of-the-art in communication systems simulation modeling; to formulate an interactive modeling/simulation concept; to determine the essential characteristics of the model; to define a flexible modeling structure and simulation process; and to demonstrate on an "intelligent" graphics display system (DEC GT-44) a small scale modeling capability developed after the conceptual model. These tasks were successfully accomplished, and the demonstrated results substantiate the practicality of a full scale implementation. This report provides much of the pertinent baseline information necessary for future development and implementation of the simulation model. This development program fits very well into the Digital Communications Simulation and Experimentation portion of TPO Number IV, "Communications for Command and Control."

Follow-on development efforts are being planned for the FY-77--FY-80 period. The information derived from this report will be used for the preparation of work statements and also as general guideline for future efforts.

Peter K. Leong

PETER K. LEONG
Project Engineer

LIST OF TABLES

Page

SECTION 2

2-1 Performance Measures 2-35

2-2 Number of Samples Required In A Monte Carlo Study 2-39

SECTION 4

4-1 Typical System Parameters and Performance Measures 4-2

4-2 Features and Options For Communication Systems 4-4

4-3 Channel and Frequency Combinations 4-10

SECTION 5

5-1 Algebraic, Trigonometric, and Function Operations 5-4

5-2 Logical Operators 5-5

5-3 Active and Passive Network Functions 5-6

5-4 Passive Network Functions 5-7

5-5 Passive Network Functions 5-8

SECTION 6

6-1 Functions of the Mathematical Library 6-8

APPENDIX A

A-1 Subroutines Used in MIDACS Demonstration Package A-3

A-2 Subroutine Directory By File Name (On DECpack DISK) . . . A-4

A-3 Display Files A-9

1. INTRODUCTION

1.1 Background

The Digital Communications Experimental Facility (DICEF) at Rome Air Development Center (RADC) is dedicated to extensive experimentation on digital communication systems. It is capable of performing test and evaluation on developmental, state-of-the-art communication equipments over real or simulated channels. To augment the capability of DICEF, a Collins C8500 Data Processing Computer had been acquired by RADC and will be installed in the near future in the DICEF. However, to effectively and efficiently utilize the processing capability of the Collins C8500 Computer in the modeling, simulation, design, analysis, and synthesis of communication systems and their components, it is necessary that appropriate interactive graphics capability be developed to effect responsive man-computer interactions. This effort is therefore intended as the initial development for the required concepts and approaches.

Traditionally, in assessing the performance of digital communications systems, designers usually resort to highly specialized studies addressing the immediate problems at hand. In most cases, the design approach places heavy reliance on analytical methods specially developed for the problem of interest. Frequently much previous work, because of lack of knowledge or the difficulty in information retrieval, is duplicated. Moreover, it is frequently impractical if not impossible to completely characterize digital communication systems using this approach. Although computers have been used to advantage, their capabilities have not been exploited to maximum benefit. Put simply, the analysis and design approach for the technology of communications has not kept pace with the technology of computers and displays and their capability for general assessment of digital communications systems, particularly in the area of high speed processing and interactive graphic displays.

Therefore, a need exists to formulate interactive modeling concepts with more flexibility and greater accuracy. The concept that can be envisaged for communications systems modeling is closely paralleled with the recent widespread use of specialized minicomputers and desk top

calculators. For example, computation of trigonometric functions is to a mathematician as computation of BER is to a communications analyst. It is apparent, of course, that the level of technology required for complete systems analysis and synthesis far exceeds that which is required for simple mathematical computation.

Thus, the method of approach in this effort was to formulate concepts for the Interactive Communications System Modeling Study that will provide answers in a meaningful manner for a wide range of digital communication systems configurations. The general approach is (1) to identify input-output characteristics of the design and modeling problem including indices of performance, (2) based on the input-output structure required, to determine the levels of modeling needed, (3) to formulate modeling concepts at the system and function level, (4) to match up the programming structure with the modeling concepts, and (5) to structure the modeling concept for interactive operation.

1.2 Scope and Objectives

The objective of this effort under Contract No. F30602-75-C-0247 was to initiate the development of an interactive communications system modeling capability for the Digital Communications Experimental Facility at Rome Air Development Center. The scope of the effort included a one year effort (1) to formulate modeling concepts for digital communications systems and to structure these concepts for interactive computer graphics, (2) to specify a flexible hierarchical computer programming structure to implement the modeling concepts and (3) to demonstrate the validity of the modeling programming structure on an RADC interactive graphic display system.

This report describes the development efforts undertaken to achieve the stated objectives. The formulated concept is described and specific recommendations are made. Demonstration software to illustrate the modeling concept is documented.

1.3 Report Organization

This report is organized into eight chapters with appendices and a bibliography including this introductory chapter. Chapter 2 reviews the formulated concept and Chapter 3 presents the programming structure. Chapter 4 describes the recommended system library; Chapter 5 describes the recommended function library; and Chapter 6 discusses hardware requirements. Software used to prepare a working demonstration is described in Chapter 7, and specific recommendations for further development are made in Chapter 8.

As a part of this effort an Interim Technical Report was prepared. Key aspects of that report are summarized in this final report; however, reference to the Interim Report will prove useful for many details.

1.4 Methodology

The approach which was followed toward the development of the interactive communication analysis model consisted of the following steps: (1) survey of the modeling literature to determine the extent of usefulness of previous approaches; (2) development and formulation of a modeling concept which has sufficient scope to accommodate the objectives of this effort; (3) investigation of the merits of mathematical approaches which will lead to a tractable and efficient model; (4) development and formulation of an interactive graphics structure which will permit interactive exercise of the model; (5) development and exercise of a demonstration interactive model to illustrate the formulated concepts; and (6) specification of a hierarchical programming structure.

For convenience in discussions and for brevity the interactive analysis model formulated under this effort will be referred to as MIDACS for Model for Interactive Design and Analysis of Communication Systems.

2. SUMMARY OF CONCEPT FORMULATION

2.1 General Considerations

The first step in developing a simulation model such as MIDACS is the identification of important modeling characteristics required. Consideration of the anticipated use of MIDACS has resulted in the identification of five important characteristics:

- Applicability
- Speed
- Accuracy
- Ease Of Use
- Expandability

The simulation model should be applicable to a broad range of problems. This means that it should have the ability to represent the operation of a large variety of communication systems. Indicative of the desired range of application are the two examples in Figure 2-1. An FSK transmitter communicating with a discriminator receiver over a cable channel is shown in Figure 2-1a, and a four phase PSK (QPSK) transmitter linked to a noisy coherent detection receiver by means of a tropo channel is illustrated in Figure 2-1b. Notice that in these two systems there is diversity in modulation form, type of channel, and type of receiving circuitry; however, structural flexibility should not be viewed as being sufficient to insure broad applicability. It is also necessary to have the ability to specify or evaluate a wide range of system parameters including such things as data rate, signal attenuation, noise level, probability of detection error, signal-to-noise ratio, channel bandwidth, etc.

The interest in a computer-supported simulation is based on the premise that a computer-oriented analysis will proceed more quickly and accurately than a similar analysis carried out manually. The validity of such an assumption is in part dependent upon the speed with which one

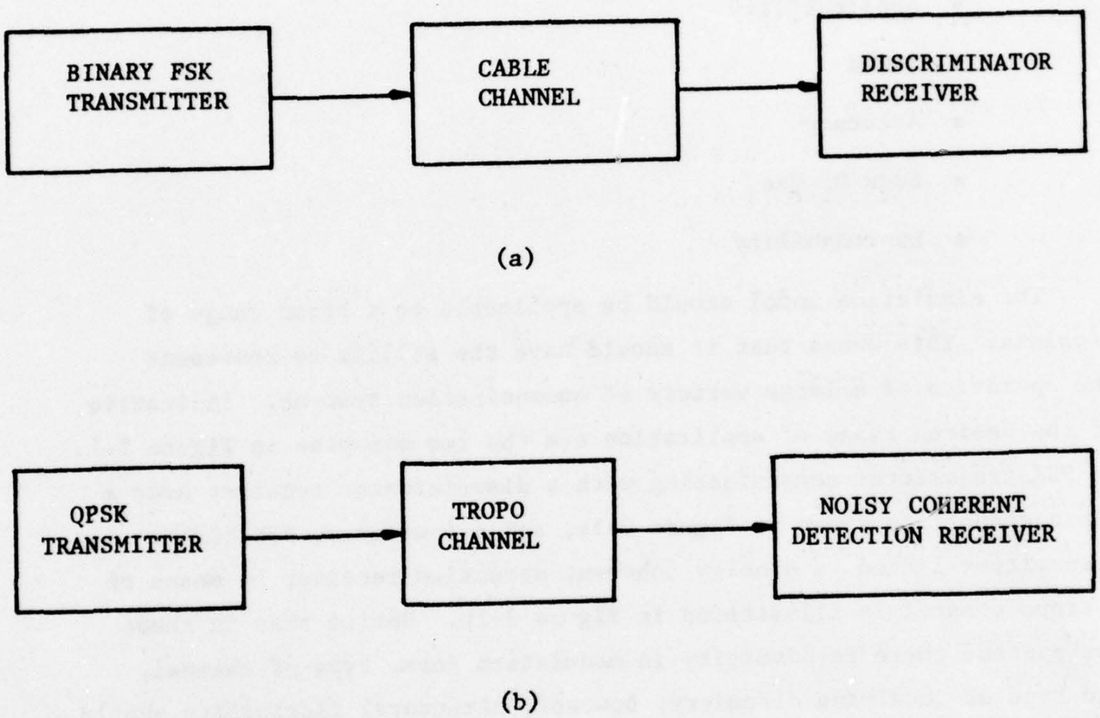
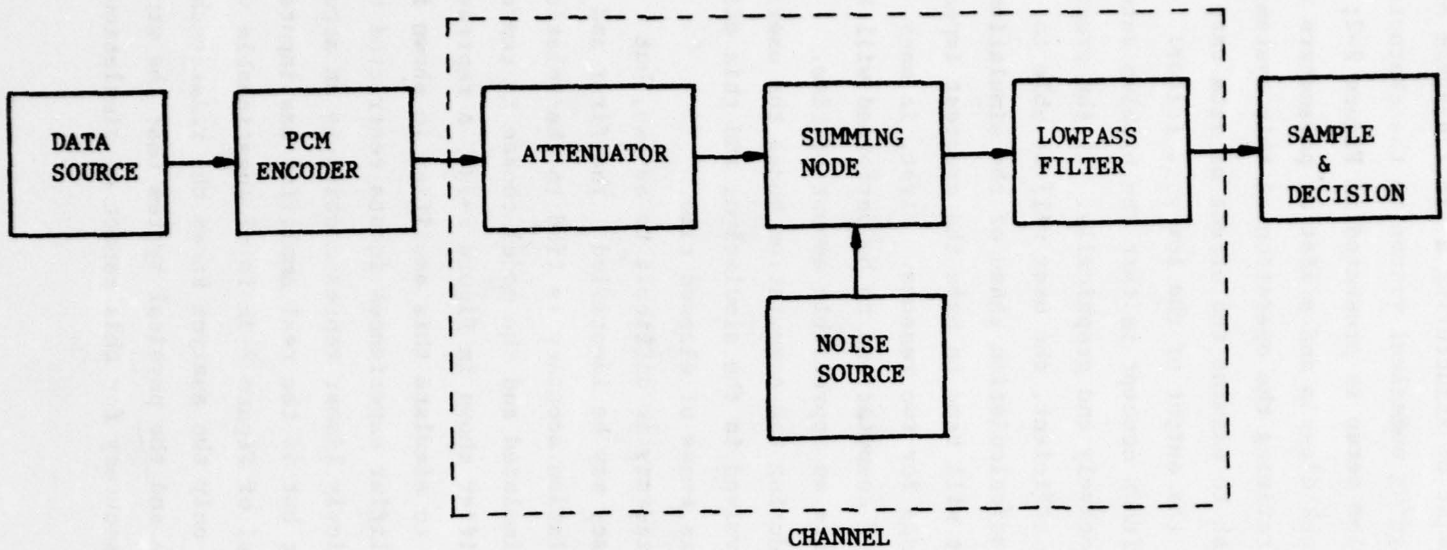


Figure 2-1. Examples of the Diversity of the Systems to Be Simulated.

can setup a problem using the computer simulation. For the communications analyst, problem setup will consist of structuring a block diagram of the system of interest and assigning numerical values to the characterizing parameters. A typical problem setup is presented in Figure 2-2; included are both the system block diagram and a list of parameters which the analyst might use in characterizing the operation of this system. For example, the analyst may wish to examine the effect of data rate on the energy-to-noise ratio at the output of the low-pass filter.

An important part of the MIDACS concept is that the problem setup will be carried out both interactively and graphically. If the programming for the interactive graphics is efficient, the user will be able to proceed quickly to the evaluation/calculation phase of the simulation. It is the calculation phase that will tend to have the greatest impact on the perceived speed of analysis for two reasons. First, in many complex simulations the number of computations to be performed will be immense and will therefore require an appreciable amount of time. Second, it is anticipated that during the computation phase the user/analyst will not be actively involved in the simulation, and this effect alone will tend to accelerate his sense of elapsed time.

The matter of simulation accuracy is difficult to assess, but three major components of accuracy may be identified. The first and most fundamental aspect of simulation accuracy is tied to the relationship between the system being simulated and the model chosen to represent that system. Consider the amplifier shown in Figure 2-3a. A representation which an analyst might use to simulate this amplifier is shown in Figure 2-3b. If the actual amplifier experiences inputs restricted to the range (-a, a), then the strictly linear representation is an accurate model for a simulation exercise; but if the real amplifier has inputs in the range (-b, b), then the model of Figure 2-3b is of questionable validity. It is important to realize that only the analyst knows the relationship between the simulation structure and the physical system that the structure represents. The assessment of accuracy for this aspect of simulation must be made by the user.



CHARACTERIZING PARAMETERS: DATA RATE
NUMBER OF ENCODED LEVELS
ATTENUATION
TYPE OF NOISE
NOISE LEVEL
TYPE OF LOWPASS FILTER
BANDWIDTH OF LOWPASS FILTER
SAMPLING TIME
DECISION THRESHOLD(s)

Figure 2-2. Problem Set Up Process for A Computer Simulation.

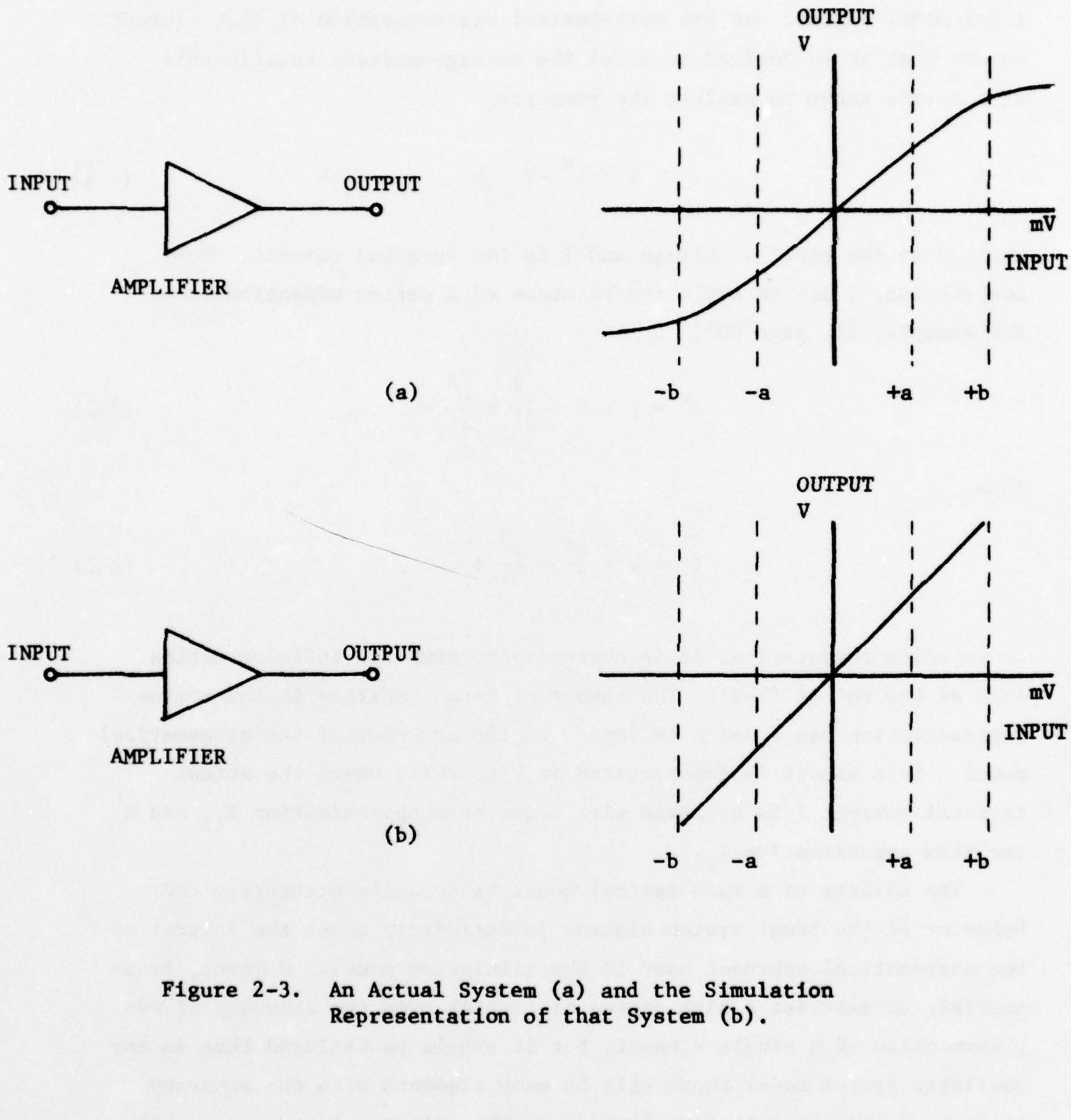


Figure 2-3. An Actual System (a) and the Simulation Representation of that System (b).

The next aspect of accuracy concerns the relationship between an ideal model element and the mathematical representation of that element. Assume that it is desired to model the voltage-current relationship of a device known to exhibit the property

$$I = e^{v} - 1 \tag{2-1}$$

where v is the applied voltage and I is the terminal current. For computation, I may be evaluated by means of a series expansion in v. For example, [1, page 985]

$$e^{v} = 1 + v + \frac{v^2}{2!} + \frac{v^3}{3!} + \ldots \tag{2-2}$$

Thus

$$I = v + \frac{v^2}{2!} + \frac{v^3}{3!} + \ldots \tag{2-3}$$

To expedite computation, it is customary to truncate infinite series such as the one in (2-3). The number of terms retained in the series representation has a definite impact on the accuracy of the mathematical model. This impact is demonstrated in Figure 2-4 where the actual terminal current I is compared with a one term approximation I_1, and a two term approximation I_2.

The ability of a mathematical model to describe accurately the behavior of the ideal system element is definitely under the control of the mathematical approach used in the simulation model. However, it is possible to exercise a high degree of control over the accuracy of representation of a single element, but it should be realized that in any realistic system model there will be many elements with the accuracy of each of the elements contributing to the output. In general, this overall accuracy is difficult to evaluate.

The third aspect of accuracy is related to the techniques employed in software development for the simulation model. When calculations are carried out manually, the analyst rarely has to worry about the order

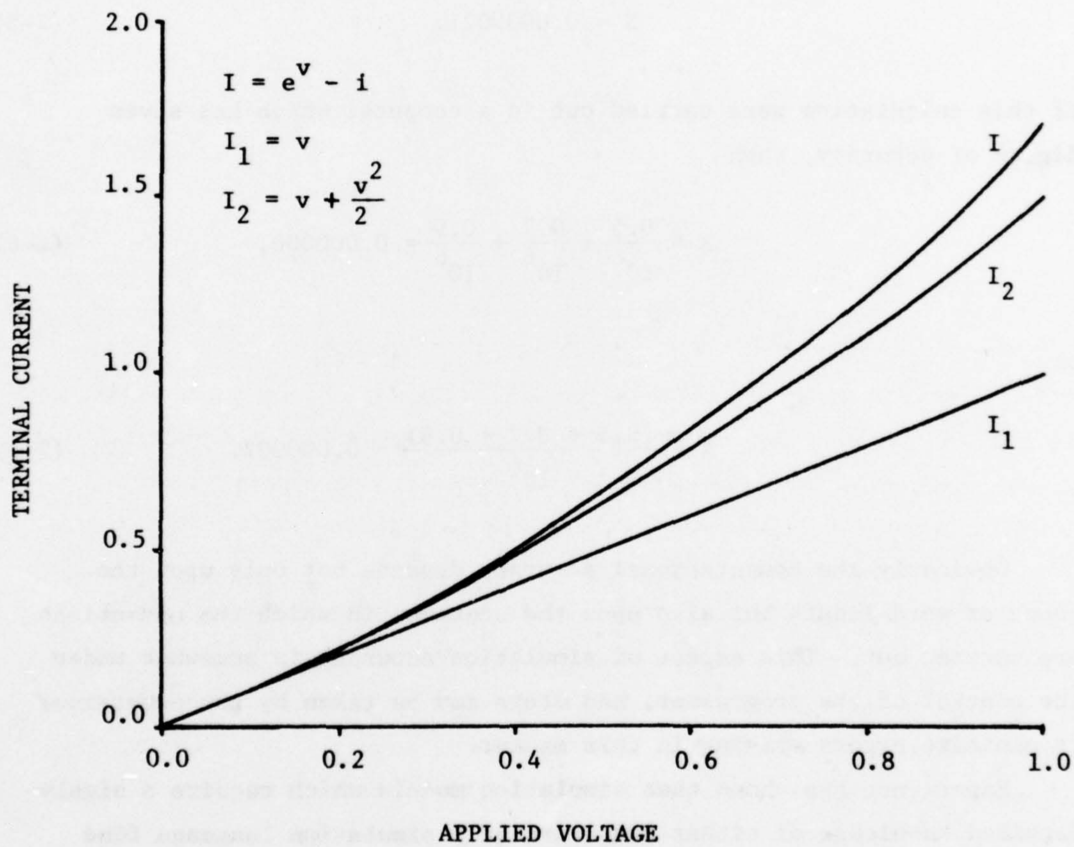


Figure 2-4. Mathematical Approximations Used to Model A Voltage-Current Relationship.

in which the basic operations such as multiplication and addition are performed since the commutative and distributive principles apply. For the computer these principles are not always applicable and can thus influence the accuracy of calculated results. Two of the more serious contributors to this problem are truncation and round-off errors which result from the finite word length of all computers. As an example, consider the following:

$$X = \frac{0.5}{10^6} + \frac{0.7}{10^6} + \frac{0.9}{10^6}\ , \tag{2-4}$$

$$X = 0.0000021. \tag{2-5}$$

If this calculation were carried out in a computer which has seven digits of accuracy, then

$$X = \frac{0.5}{10^6} + \frac{0.7}{10^6} + \frac{0.9}{10^6} = 0.000000, \tag{2-6}$$

or

$$X = \frac{(0.5 + 0.7 + 0.9)}{10^6} = 0.000002. \tag{2-8}$$

Obviously the computational accuracy depends not only upon the computer word length but also upon the sequence in which the operations are carried out. This aspect of simulation accuracy is somewhat under the control of the programmer, and steps may be taken by the programmer to minimize errors arising in this manner.

Experience has shown that simulation models which require a highly-detailed knowledge of either computers or a simulation language find little use. Since it is desired that MIDACS gain wide acceptance, by communication analysts, it is imperative that the model be easy to use.

Ideally, the inexperienced analyst should be able to exercise the model with little or no external assistance, and at the same time the experienced user should not be subjected to lengthy or detailed descriptions of basic operations. Thus, the program should be highly interactive, but it should be flexible in the amount of descriptive detail provided to the user.

Since any simulation model can contain only a finite library of simulation elements, and since it is impossible to anticipate every structural element desired by future program users, it is desirable to structure the program so that it can be expanded easily by the addition of new or alternate simulation elements. This approach allows the user to create representations which may be peculiar to his own interests, and it also allows the rapid addition of new communications components to the simulation element library.

Five important modeling characteristics have been identified and discussed. It is now necessary to indicate how each of those characteristics may be incorporated into a modeling structure.

In the development of simulation concepts it has been assumed that a system model will be structured by the analyst from a set of models of basic digital communication functions. Such a structure is indicated in the example of Figure 2-2. For a modeling structure of this type three things must be done to insure that the overall simulation technique is broadly applicable. First, the library of models of basic communication functions must be extensive. This means that not only should there be a wide variety of different functions in the library, but there should also be a variety of implementations of the same function. Second, each function model should contain sufficient detail to allow specification by the analyst of the important parameters affecting functional performance. For example, a user on identifying the need for a low-pass filter model should be able to specify filter type, order, ripple, bandwidth, etc. Third, the program should be capable of evaluating the different performance parameters of greatest importance and widest use. It should be emphasized, however, that this does not preclude the introduction of new performance measures which either facilitate the evaluation process or enhance the understanding of system operation.

Two contributors to the time required to perform a simulation have been identified: the time required to structure the simulation model and the time required to exercise the model to obtain the desired data. Consider first the problem of model structuring. Many of the existing or planned digital communication systems conform structurally to one of a limited number of well-documented system models, and therefore the design or analysis of such systems involves only the exercise of one such common model with parametric variations. In such instances time may be saved by providing the user with a complete system model. The system model would describe a fixed system structure and would allow parametric specification in the manner employed for function models.

It is easy to envision at least one other situation in which a system model would be desirable. Consider the case where an analyst is examining new system structures. In the process of doing so he may wish to re-examine the performance of a particular system several times. This process would be very lengthy if each system exercise required a complete restructuring of the system model from function models; however, it would proceed quite rapidly if the initial system structure could be retained and identified as a system model in the simulation program's library.

The second factor contributing to the speed of the analysis is computation time. In many instances complete analytical studies have been done on a given system structure, and therefore, carrying out a true simulation of such systems would waste a great deal of time. The incorporation of the analytical forms as system level models in the program would provide the analyst with the means of obtaining performance data on a wide array of the more commonly used digital communications systems. For those instances where an actual simulation is to be performed, computation time may be minimized by providing a multilevel representation of the functional blocks. The levels of representation would correspond to the amount of detail included in the functional model, and it is assumed that the evaluation of a complex and/or detailed representation would require greater computation time than would the evaluation of a simpler representation.

It has been observed that the accuracy of the computed results depends upon the structure of the simulation model, the mathematical representations employed for the individual functional elements, and the computational techniques employed. The degree to which the simulation model accurately reflects the structure of the physical system being simulated must be the responsibility of the analyst. No software program can reliably anticipate the intent of the user in structural matters.

The concept of a multilevel representation for the functional elements has been noted above in the discussion of computational speed. The relationship between computation time, accuracy, and the multilevel structure is shown in Figure 2-5. The lower bound of error, which is defined to be the square of the difference between the calculated value $\hat{x}$ and the true value x, is seen to decrease as the level of representation is increased. In this context, level 1 is the most general and level 3 is the most detailed representation. Furthermore, it should be observed that for a fixed computation time the higher level model does not assure a higher degree of accuracy. The point being made is this: accuracy, speed, and computation time are intricately interrelated in such a manner as to preclude any a-priori absolute specification of any one characteristic. Trends, however, such as those indicated in Figure 2-5 are discernible and are useful in the development of program concepts.

The degree to which MIDACS is perceived to be easy to use is in major part dependent upon the user. A user with little experience in the use of simulation programs or knowledge of communication systems will tend to prefer a program requiring little advanced knowledge of the program details and a thorough, preferably interactive, set of instructions. The advanced user tends to minimize the need for step-by-step instructions and to emphasize the ability to quickly setup and analyze a problem, the ability to quickly and easily alter the system structure or parameters, and the ability to address a wide range of problems.

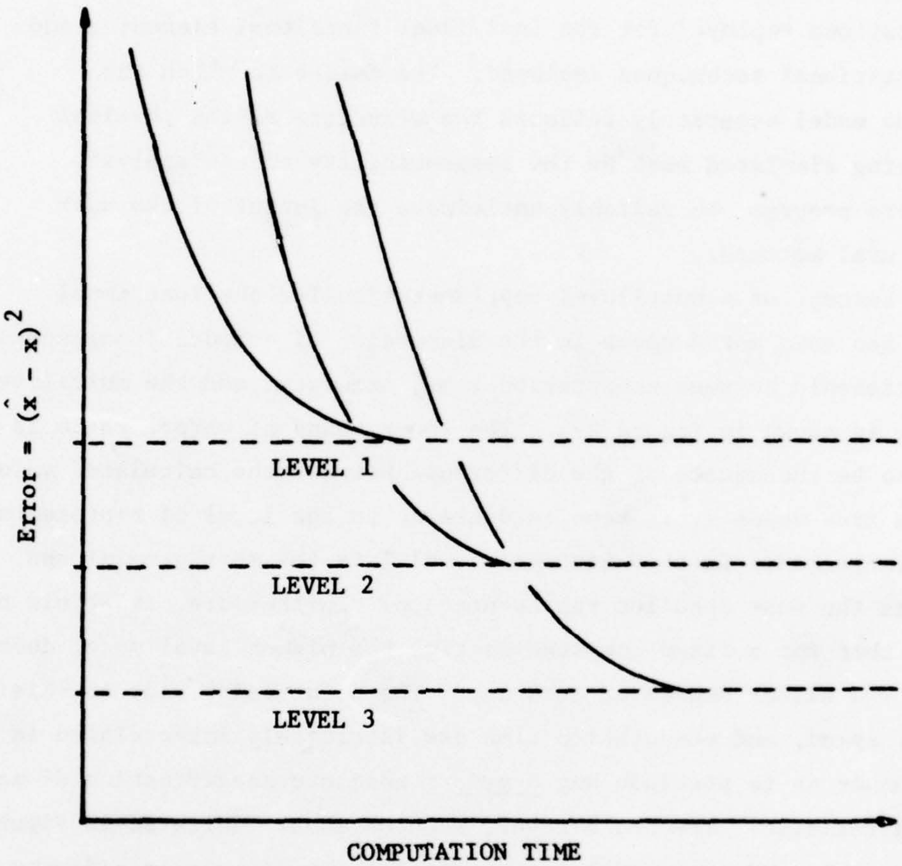


Figure 2-5. The Relationship Between Level of Analysis, Computation Time, and Error.

The needs of both types of users can be met by a single modeling structure provided that the program structure recognizes the different needs. An example of an appropriate program structure is given in Figure 2-6. On starting the program, the user is presented text describing the program's operation and how to respond to questions posed by the program. Next, the user is asked if he needs instructions on program use. At this point the advanced user may respond NO and proceed directly to setup his problem. If the answer is YES, the program will provide step-by-step instructions and explanations for the user.

There are a variety of ways that a program can be written so that future expansions or modifications can be made easily. One of the better approaches is to modularize the program so that each program element or operation is an identifiable entity, such as a subroutine. This step should be supplemented by the development of a complete set of program documentation describing the interrelations of the various program elements.

At this point, some points of terminology should be discussed. The simulation model that has been described thus far may in general be characterized as a multilevel program. The program has two levels of modeling structure: a system level and a function level. The system level structure is fixed and is not alterable by the user, but the function level structure is totally determined by the analyst during the problem setup phase of the simulation. Also, for many of the models there will be two or more representations of a given function or system representing two or more levels of modeling complexity. As noted previously, the multilevel mathematical representations constitute a trade-off between modeling detail and computation speed.

Another matter of terminology that should be clarified is how the models are described. Function models present no problem since the function being performed seems to be the best model descriptor, but the system models present an entirely different situation. A block title should be short, and at the same time it should relate the most significant characteristics of the block. Consider the structure presented in Figure 2-7. A complete description of this system would have to identify in some

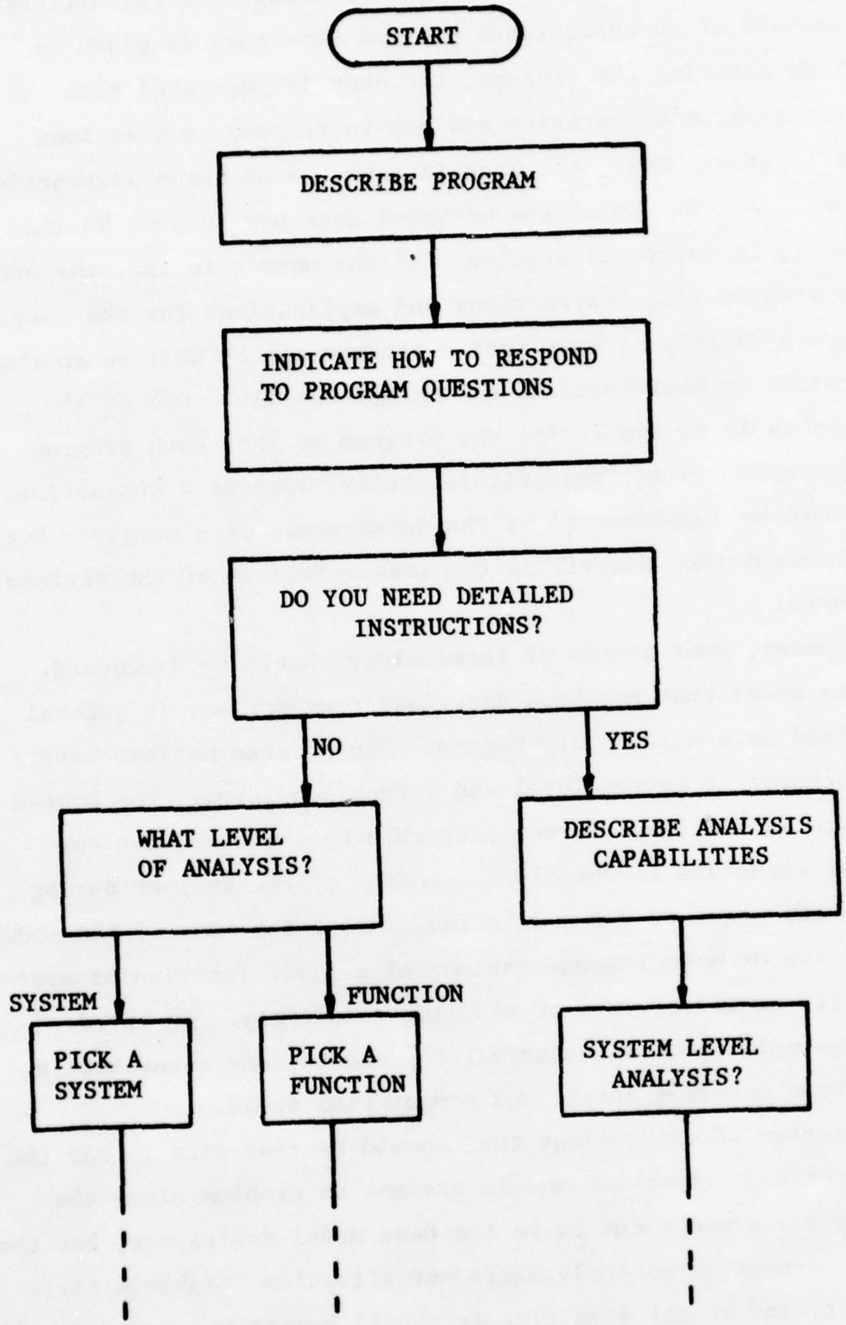


Figure 2-6. Flow Chart for A Highly Interactive Programming Structure.

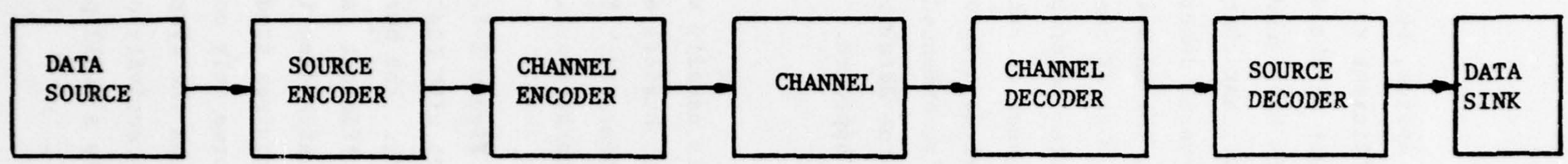


Figure 2-7. Basic Components of A Communication System.

detail each of the blocks comprising the system, but if a short descriptive label is needed it may be sufficient to identify only that one block that has the greatest impact on the system as a whole.

Historically, communications analysts have tended to identify the waveform modulation as the most significant characteristic of a system. However, there are alternative system descriptors which are sometimes used as primary adjectives, for example, "line of sight microwave", "tropo link", or "HF link". Most of the work on MIDACS to this point has used as the primary system descriptor the waveform modulation, e.g., PSK, QPSK, FSK, etc. Based on recent discussions with RADC and DICEF personnel, the channel descriptor may be more readily understood by MIDACS users. The recommended modeling structure can accommodate either approach although the detailed structured hierarchy must be adapted to the choosen approach.

2.2 Modeling Concepts

A model in the engineering context is usually a set of mathematical equations which describe some aspect of a physical entity which either exists or is in the process of being implemented. This section of the report will describe the important characteristics of such models and how such models are developed.

Consider the flowchart presented in Figure 2-8. This figure indicates that the modeling process begins with the identification of the particular physical system of interest. The next step in the process requires that the analyst have sufficient familiarity with the given system to produce a block diagram which describes accurately that aspect of the system's operation which requires study. In some instances the system may be so simple that it performs only one task, and therefore, the block diagram would completely describe the system's operation. However, in most realistic situations the activity of interest and thus the related block diagram constitutes only a small part of the system's total operation.

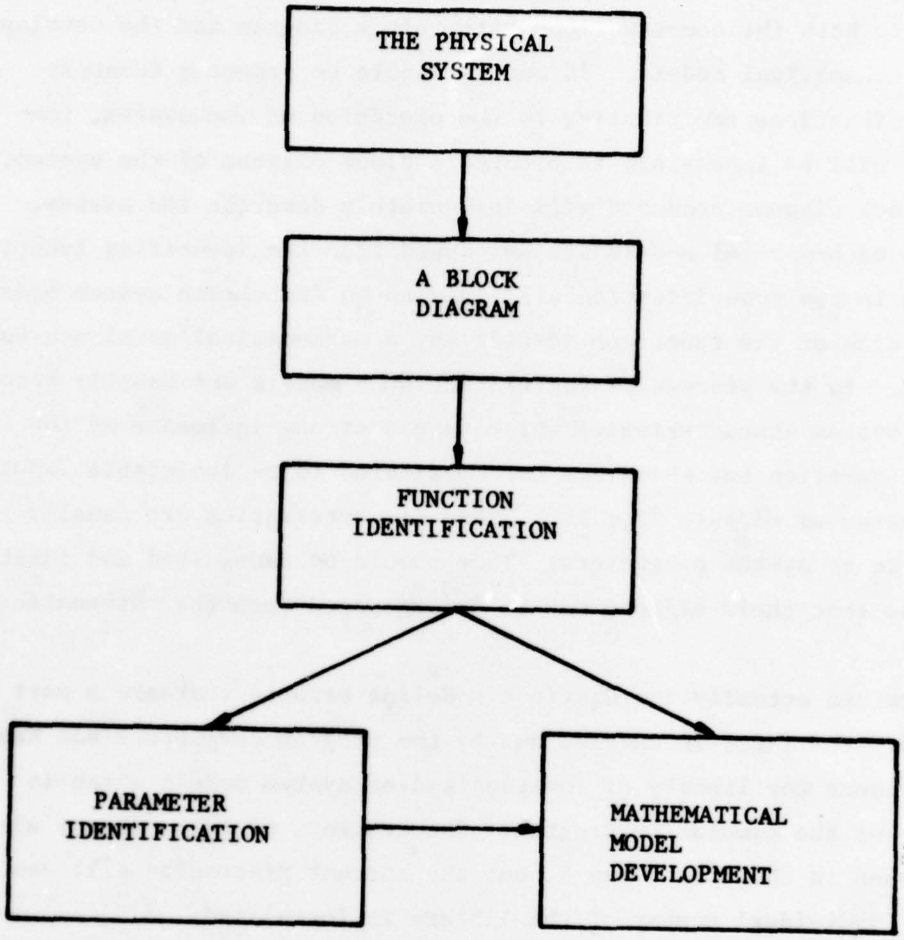


Figure 2-8. The Modeling Process.

In the process of producing a block diagram, the analyst has been required to isolate and identify those basic functions which contribute to the overall operation of the system. This identification process is critical to both the construction of the block diagram and the development of mathematical models. If one is unable to properly identify the basic functions contributing to the operation of the system, then either it will be impossible to produce a block diagram of the system, or the block diagram produced will inaccurately describe the system. Since the mathematical models are developed from the identified functions, any error in the identification will lead to an inaccurate system model.

For each of the functions identified, a mathematical model may be developed. In the process of formulating such models one usually becomes aware of system characteristics which have a strong influence on the system's operation but which are not considered to be isolatable inputs to the system or outputs from it. These characteristics are usually referred to as system parameters. They should be recognized and identified clearly so that their effects may be incorporated into the mathematical model.

There are actually two distinct modeling efforts that are a part of MIDACS. The first is carried out by the program developers and has as an end product the library of function and/or system models which is the heart of the simulation program. The contents of this library will be discussed in Chapters 4 and 5, but the current discussion will center on how an individual member of the library is formulated.

As indicated in Figure 2-8, the first step in the modeling process is the identification of the physical system to be modeled. For MIDACS, the scope of interest is actually a class of systems: digital communication systems. Next, a block diagram should be produced, for each specific system of interest, and the functional blocks which are a part of that diagram should be identified. The next step is identified in Figure 2-9, and it requires the developer to specify the type of model desired: system or function. It should be emphasized that this option is a direct result of the multilevel modeling concept in MIDACS. Assume for the moment that a system level model is desired.

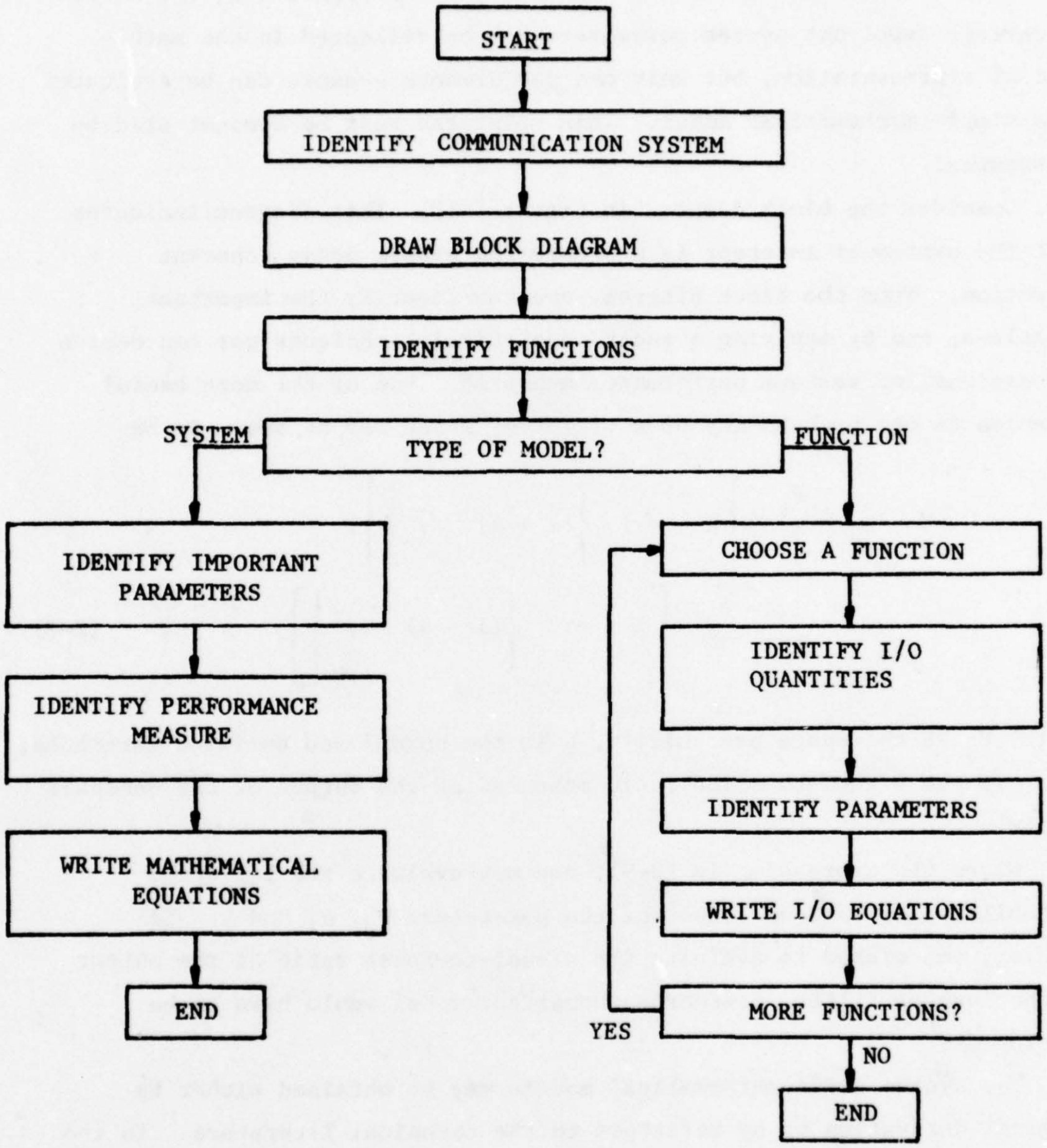


Figure 2-9. The Development of System and Function Level Models.

The system level model for MIDACS is a mathematical representation intended to describe one performance measure in the environment of one particular communication system. As indicated in Figure 2-9, the effect of certain important system parameters can be reflected in the mathematical representation, but only one performance measure can be evaluated by a single mathematical model. This point can best be demonstrated by an example.

Consider the block diagram in Figure 2-10. This diagram indicates that the system of interest is a binary PSK system using coherent detection. From the block diagram, one can identify the important functions, and by applying standard analytical techniques one can derive expressions for various performance measures. One of the more useful measures is the probability of a bit error which may be shown to be

$$P_e = \frac{P_s}{2} \left[1 - \mathrm{erf} \left\{ (1 + \alpha) \quad \sqrt{\gamma} \right\} \right] + \frac{(1 - P_s)}{2} \left[1 - \mathrm{erf} \left\{ (1 - \alpha) \quad \sqrt{\gamma} \right\} \right], \tag{2-9}$$

where, P_s is the space probability, α is the normalized decision threshold, and γ is the signal-to-noise ratio measured at the output of the bandpass filter.

Given the expression in (2-9), one may evaluate the bit error probability for various values of the parameters P_s, α, and γ. If however, one wished to evaluate the signal-to-noise ratio at the output of the lowpass filter, another mathematical model would have to be developed.

The system level mathematical models may be obtained either by original derivation or by reference to the technical literature. In the former case a skilled communications engineer will be required to generate the required block diagrams and the related mathematical descriptions. The latter alternative is somewhat more efficient since previous results can be utilized. In particular, many models are readily available from the following sources: [2], [3], [4], [5], [6], and [7].

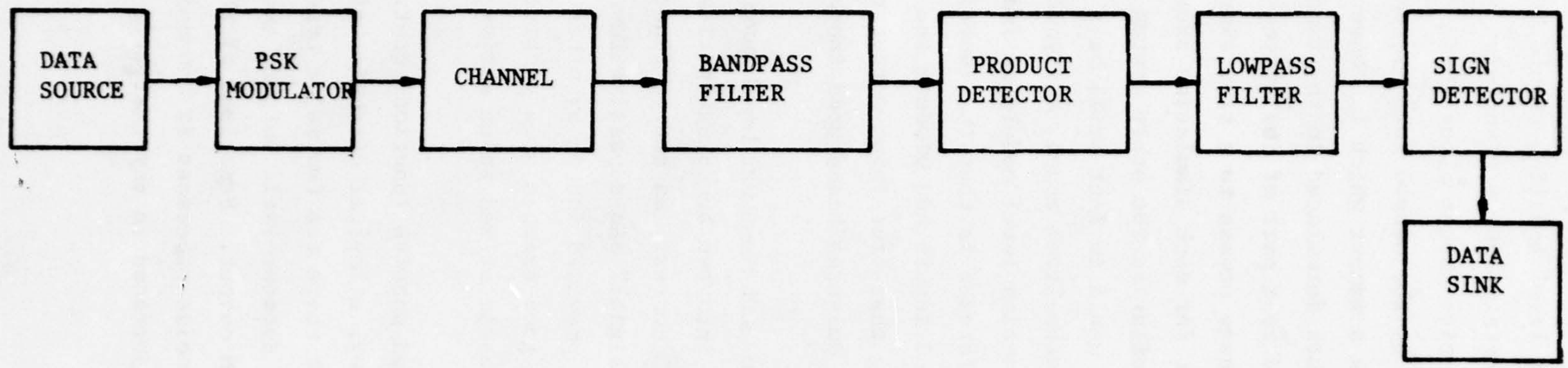


Figure 2-10. The Block Diagram for a System Level Model of A Binary Coherent PSK System.

One additional approach to system level modeling should be considered. It is conceivable that certain system structures will find frequent use and that the information sought about these systems may exceed that which can be supplied by the closed form system models described above. To meet this need in a manner which is convenient for the analyst, one could allow a structure formulated in the function-level mode of operation to be retained as a part of the system-level library. In this way the user could have access to a true simulation model without having to reconstruct it for each simulation exercise. Although such a model would appear similar to the other system models using closed form representations, it would in fact still be a function-level structure. In contrast to a system-level model, a significantly different approach is required for function-level models as indicated by Figure 2-9. The most significant difference is that the function-level models are designed to accept simulated inputs and produce the corresponding outputs. It bears emphasizing that true input-output relationships are involved and not just performance measures based upon those inputs or outputs.

Another difference between system and function-level models is that the functions being modeled fall into two broad categories: active and passive functions. In the present context, an active function is defined to be one which generates a "signal" and a passive function is one which simply operates on a signal provided for it by other functional blocks. A low-pass filter would be a good example of a passive function, while a waveform generator would be characterized as an active functional element.

The distinction between active and passive functions contributes little to the modeling process itself; however, a logical extension of this classification is the recognition that there are two major types of active functions. The first type produces a deterministic output, and the second type produces a random or pseudorandom output. Functions of the latter type are typically used to represent noise processes in communication systems and as such are of extreme importance in any realistic modeling effort.

It has been noted that the mathematical models for functional blocks actually describe an input-output relationship. Such relationships could be developed in a variety of ways, and given a little imagination one could envision a rather broad range of possible input-output pairs. For MIDACS, a somewhat traditional approach has been taken, and inputs are generally assumed to be time domain "signals", and outputs are those time domain responses to the specified outputs.

The selection of time domain modeling bears some discussion. It is recognized that it is possible to carry out much of the modeling and simulation effort in the frequency domain (particularly for linear functions), and furthermore, it is acknowledged that there are several points in the typical communication system where frequency domain information may be of greater interest. However, there are several critical points in the typical digital communication system where time domain information is essential for assessment of system performance. One such point is the output of the data source. Rarely is this point characterized by its spectrum. A similar situation exists in the receiver's detection/decision circuitry where the need for time domain information at this point is emphasized by timing circuitry associated with the decision circuits. Nonlinear functional elements, as a class, are most easily modeled in the time domain. It may be concluded that the most efficient modeling approach lies in the time domain since several important points in the signal path have critical ties with time domain information. This should not be viewed as a disregard of the need for spectral information, since it is recommended that a fast Fourier transform (FFT) algorithm be made a part of MIDACS. Thus, when spectral information is needed, the analyst may transform the time-domain data to the frequency domain.

The final topic of this section is the actual implementation of mathematical models for functional blocks. The area of system analysis has given rise to a variety of forms for describing the operation of physical systems, and four techniques will be described.

Consider the functional elements depicted in Figure 2-11. Each functional element is commonly encountered in the block diagrams of digital communication systems, and each element is described by an algebraic equation. Note that both linear and nonlinear functions may be described by algebraic equations. The major shortcomming of the algebraic representation is that it does not account for element memory, but in recognition of the multilevel modeling concept it should be emphasized that a no-memory representation may be sufficient as an approximation in some cases.

A second approach is to represent a function by an ordinary differential equation, and then to seek a solution to that equation. The study of differential equations has a long history, and solutions have been found for many of the more commonly-encountered forms; however, the analytical solutions for these equations often have limited appeal for computer applications. The basis for this will be illustrated by an example.

Consider the simple lowpass filter of Figure 2-12(a). This circuit may be described by a differential equation by standard techniques illustrated in texts on network analysis and mathematics: [8, page 114], [9, page 72]. The result of such an exercise is the equation appearing in Figure 2-12(b). This equation is a linear first order ordinary differential equation, and the solution for the output voltage $V_o(t)$ in terms of the input voltage $V_i(t)$ and the filter elements R and C is given by [10, page 185].

$$V_o(t) = e^{-h} \left[\int_{\tau}^{t} e^{h} \, V_i(t) \, dt + K \right] \tag{2-10}$$

where

$$h = \int_{\tau}^{t} \frac{dt}{RC} = (t - \tau)/RC \tag{2-11}$$

and K is a constant determined by initial conditions at time τ.

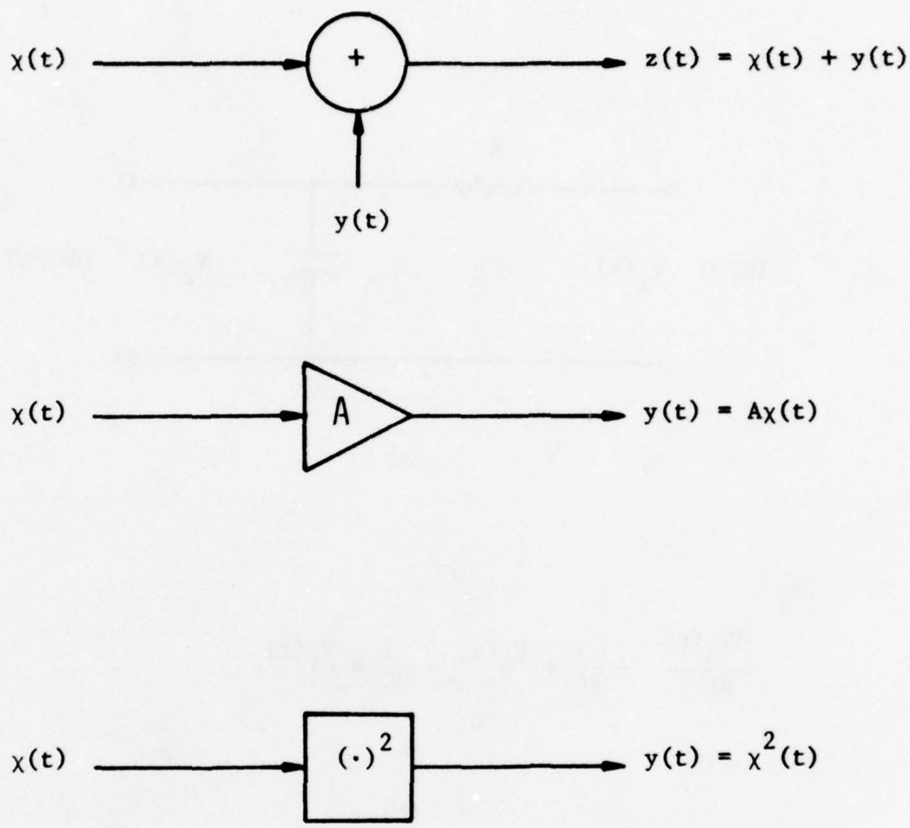
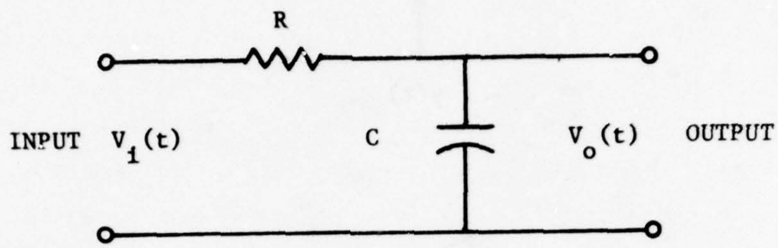


Figure 2-11. Functions Described By Algebraic Equations.



(a)

$$\frac{dV_o(t)}{dt} + \frac{1}{RC} \bullet V_o(t) = \frac{1}{RC} \bullet V_i(t)$$

(b)

Figure 2-12. (a) A Lowpass Filter and,

(b) Its Differential Equation Representation.

Equation (2-10) accounts for both the transient and steady-state components of the output voltage, but the question remains as to how to evaluate this equation. One possible approach is to use the techniques of numerical quadrature [11, page 59] to evaluate the integral in (2-10), but all that would be accomplished by doing so is that numerical integration would be substituted for a numerical solution to the differential equation.

One might consider the possibility of analytically performing the integration in (2-10) with the hope that the final result will be a simple algebraic expression for $V_o(t)$, but that approach has two major problems. First, there is no guarantee that a closed-form analytical solution exists for an arbitrary $V_i(t)$. Second, even if analytical integration is possible, the result obtained will apply only to the particular form of $V_i(t)$ assumed for the integration. If $V_i(t)$ changes, another output expression is required.

Another approach uses the standard techniques for numerically solving differential equations. This approach has been elegantly developed by state variable analysis for time variant and nonlinear systems [12, page 313], [13, page 114], [14, page 60].

In the state variable approach it is customary to describe a system by a pair of vector-matrix equations [13, page 107].

$$[Q(t)] = [A][Q(t)] + [B][X(t)] \tag{2-12}$$

$$[Y(t)] = [C][Q(t)] + [D][X(t)] \tag{2-13}$$

In the above equations, [X(t)] is the input vector, [Q(t)] is the state vector, and [Y(t)] is the output vector. Notice that equation (2-12) is in the form of a first order differential equation and that it is necessary to solve the equation in order to evaluate the output vector [Y(t)]. In theory this method allows one to interface multiple blocks by performing matrix multiplication operations; however, a practical limit is reached very quickly for this interfacing approach. The limit is established by the maximum array size permitted for the host computer and the excess computation time required in inverting one large matrix rather than several smaller ones.

There are a variety of numerical techniques available for solving differential equations [11, page 73]. The general concept will be illustrated by employing one particular approach: the Runge-Kutta method. The differential equation appearing in Figure 2-12(b) will be used, but it will be rewritten in the form

$$\dot{V}_o(t) = f[t, V_o(t)] \tag{2-14}$$

where

$$f[t, V_o(t)] = \frac{V_i(t) - V_o(t)}{RC} \quad . \tag{2-15}$$

By applying the Runge-Kutta method, successive sample values of $V_o(t)$ may be determined by the following equation

$$V_o(t_{n+1}) \simeq V_o(t_n) + \left(\frac{T}{2}\right) f\,[t_n, V_o(t_n)] + \frac{T}{2}\; f\,[(t_n + T), V_o(t_n) + Tf[t_n, V_o(t_n)]]. \tag{2-16}$$

In the above, t_n is the n-th sampling time, and T is the time between samples.

The result of substituting equations (2-14) and (2-15) into (2-16) is

$$V_o(t_{n+1}) \simeq V_o(t_n) + \left(\frac{T}{2}\right) \frac{V_i(t_n) - V_o(t_n)}{RC} + \frac{T}{2}\; \frac{V_i(t_n+1) - V_o(t_n) - \frac{T}{RC}\,[V_i(t_n) - V_o(t_n)]}{RC} \quad . \tag{2-17}$$

Equation (2-17) could be used in this form to evaluate the output voltage $V_o(t)$ in terms of the input voltage $V_i(t)$ and the RC time constant.

The final mathematical modeling approach to be discussed was developed expressly for the simulation of continuous time systems by a digital computer. The technique leads to a difference equation similar to the one of (2-17), but this difference equation is obtained by use of the Z-transform. A detailed discussion of this approach is presented by Rosko [14, page 85], and the following material will summarize Rosko's presentation.

All signals in a digital simulation are represented by discrete time samples of the actual continuous time signals. This sampling process tends to alter the nature of the involved signals which is easily demonstrated in the frequency domain. Figure 2-13 shows how the signal's spectrum is affected. Sampling at or above the Nyquist rate does the least harm to the signal, and sampling below the Nyquist rate does irreparable damage to the signal. Compensation can be made for the alteration in the signal caused by sampling at or above the Nyquist rate by introducing what Rosko [14, page 91] refers to as data reconstructors into the functional representation. Figure 2-14 shows the relationship between a given functional block and its representation including the data reconstructor.

The type of data reconstructor to be employed in a given situation is somewhat dependent upon the input waveforms that may be anticipated. In particular, the reconstructor used for a given model should as accurately as possible reflect the input signal's trajectory between the samples taken of that signal. In many digital systems, the waveforms are rectangular, and therefore, a zero-order hold device makes an excellent data reconstructor. The mathematical model that will be developed will thus actually describe the function appearing in Figure 2-14(b).

As an example consider the circuit of Figure 2-12. The starting point for this method is the circuit transfer function which is:

$$G(s) = \frac{\alpha}{s+\alpha} \tag{2-18}$$

$$\alpha = 1/RC \tag{2-19}$$

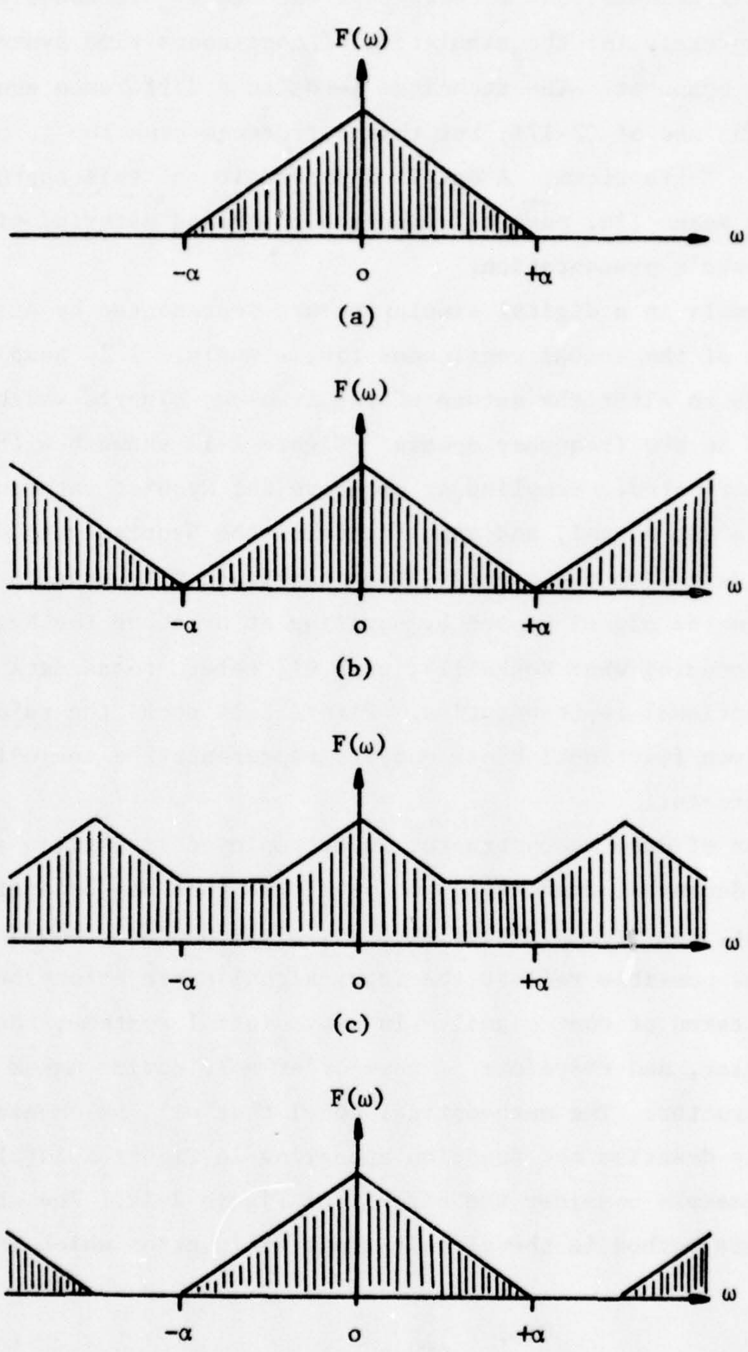


Figure 2-13. A Signal's Spectrum (a), When Sampled At The Nyquist Rate (b), When Sampled Below the Nyquist Rate (c), and When Sampled Above the Nyquist Rate (d).

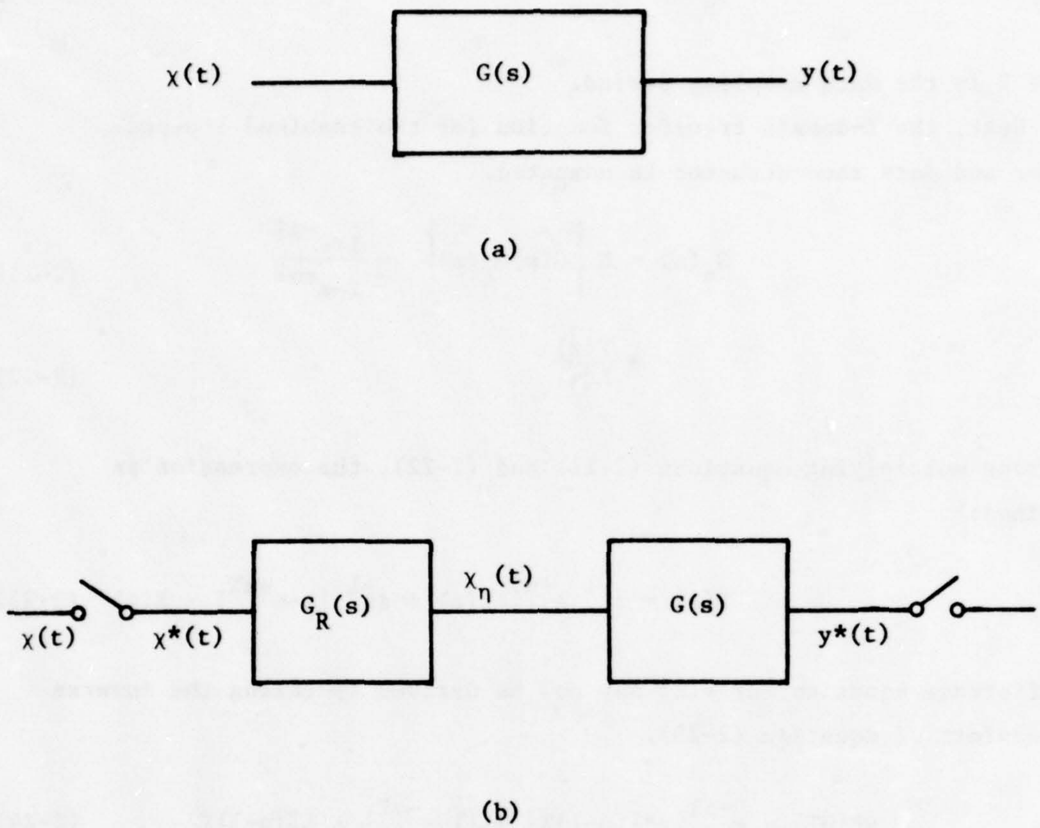


Figure 2-14. A Functional Block (a) and Its Model Using A Data Reconstructor(b).

The data reconstructor is assumed to be a zero-order hold, and its transfer function is

$$G_R(s) = \frac{1-e^{-Ts}}{s} \tag{2-20}$$

where T is the data sampling period.

Next, the Z-domain transfer function for the combined low-pass filter and data reconstructor is computed.

$$G_s(z) = Z\left\{G(s)G_R(s)\right\} = \frac{1-e^{-\alpha T}}{Z-e^{-\alpha T}} \tag{2-21}$$

$$= \frac{Y(z)}{X(z)} \tag{2-22}$$

By cross multiplying equations (2-21) and (2-22), the expression is obtained:

$$Y(z) = z^{-1}\ e^{-\alpha T}\ Y(z) + z^{-1}\ [1-e^{-\alpha T}] \cdot X(z) \tag{2-23}$$

A difference equation for y(t) may now be derived by taking the inverse Z-transform of equation (2-23).

$$y^*(nT) = e^{-\alpha T}\ y^*[(n-1)T] + [1-e^{-\alpha T}] \cdot X[(n-1)T] \tag{2-24}$$

One important characteristic of this result should be pointed out. For an arbitrary input, x(t), this representation is an approximation, but for rectangular input waveforms it is an exact solution.

Several mathematical modeling techniques have been examined, and their respective strengths and weaknesses have been pointed out. It might be expected that one approach should be singled out as being the approach to use; however, this will not be done since no one approach is superior in all circumstances. Instead, it is recommended that each model be considered individually and that the unique features of each function be used to determine which technique is most suitable. This approach is feasible since the MIDACS concept requires only the compatibility of time-domain inputs and outputs.

One additional modeling concept that should be noted is the modeling of signals and functions in the RF portions of communication systems. It is readily observed that in most communication systems the carrier component of the transmitted or received signal conveys little or no information itself, and therefore, the simulation of such a component would add little to an analysis. A similar situation exists for RF circuit elements such as bandpass filters. Rarely is any unique feature associated with a given center frequency for a bandpass filter. The normal point of interest is the filter's effect upon the spectrum of the signal's intelligence. For narrowband systems it is possible to generate a lowpass or baseband model of the traditional RF signals and circuits [15].

The baseband representation of both signals and circuits is desirable since it will significantly reduce the number of samples required for a simulation. Manske [15] shows that such a representation is possible if the narrowband assumption is valid and if the in-phase and quadrature components of the signals are retained. He then proceeds to show how models for a variety of circuit functions can be developed using these in-phase modulators and demodulators, and frequency-independent non-linearities. The details of how the various elements can be modeled will not be repeated here since the intent of this effort is to determine workable concepts rather than to produce specific models.

Earlier it was indicated that there are two aspects of modeling in the MIDACS concept. The first aspect, model development, has been discussed above, and the second aspect, model use, will now be considered.

A user of MIDACS would draw upon the library of mathematical models developed by application of the concepts described above. However, it is important to distinguish between the highly detailed and complex task of mathematical model development and the somewhat simpler task of using the models. Model use is actually a software/programming consideration since it is concerned primarily with what models are available, what the characteristics of a given model are, how a given model is connected to another model to form a system structure, what mathematical techniques are employed to exercise the models, and what models are compatible. This aspect of modeling will be discussed in Chapter 3.

2.3 Simulation Concepts

As discussed here, simulation is a multistep process consisting of:

1. The identification of the system of interest,

2. The structuring of a block diagram of that system which employs the mathematical models previously discussed,

3. The specification of measures of performance which are to be evaluated during the simulation exercise, and

4. The exercising of the model for evaluation purposes.

Items one and two of the above are a part of the modeling concept presented in the latter part of Section 2.2, and will not be discussed further here.

The primary objective of a system simulation is the evaluation of some aspect of a system's performance. This evaluation is usually carried out by defining certain performance measures and then determining a quanitative value for that measure in the system being studied. A variety of performance measures have been identified and defined in the technical literature on digital communication systems, and since many of the measures have gained widespread acceptance, it is logical to incorporate them into the MIDACS structure. Some of the more important measures are identified and discussed in Table 2-1.

For any performance measure of interest, a software program will be required which (1) indicates how the system model is to be exercised, (2) identifies what data is to be collected for evaluation purposes; and (3) defines how the collected data is to be used in obtaining a quantitative value for the measure of interest. As an example, assume that spectral information is desired. The appropriate subroutine would proceed to process a typical data waveform through the system. At the point where spectral information is required, a collection of time domain samples would be taken. The collected data would then be used in a fast Fourier transform (FFT) algorithm to produce the required spectrum.

TABLE 2-1

Performance Parameters

NAME	SYMBOL	GENERAL DESCRIPTION
Probability of Error	P_e	Indicates the likelihood of a receiver detection error
Signal-to-Noise Ratio	SNR	Describes the relative magnitude of signal and noise power components
ISI Ratio	-	Measures the degree to which bandwidth limitations are affecting received signal waveform and the detection characteristics
Spectrum	-	Shows the relative importance of the various signal/noise frequency components and may be used to determine bandwidth
Decision Variable Distribution	-	Indicates the probability of the decision variable assuming certain values

In most instances the probability of error (P_e) parameter is considered to be of the greatest importance, and for this reason several exercise techniques have been explored which allow the evaluation of P_e.

The first exercise technique to be discussed will be referred to as the analytical/numerical approach. This exercise form is primarily intended for system level models and does little more than evaluate the appropriate system mathematical model, given the current parameter specifications.

The most serious shortcomings of this approach are that it is not generally applicable, and that a skilled analyst may be required to select required functional expressions relating parameters to performance measures.

Requirements for applicability of this method, in the typical case, are systems whose components are linear, noise which is additive and Gaussian, and interference which is additive. Both systems without memory and systems with memory can be handled, but the latter lead to less tractable expressions.

Typical effects which are more difficult to handle using this method are: nonlinearities, nonadditive noise, and non-Gaussian noise.

Among the advantages of this method is the fact that it can be very efficient computationally, especially if the final functional expression is in closed form. Of course, more computer time is required if numerical methods must be used to evaluate integrals, solve differential equations, or approximate the sums of infinite series. On a relative basis, however, almost any analysis using the analytical/numerical approach requires less computer time than alternative methods.

Next, a group of exercise approaches will be discussed which lead to the more traditional simulation. As a group, these techniques fall into the category of Monte Carlo studies since they in theory subject the system to all possible combinations of inputs.

The most straightforward "Monte Carlo" approach would use exclusively deterministic inputs. Thus, for fixed inputs and parameters the function outputs and the final system output would be deterministic functions of time. This type of Monte Carlo study would apply to communication systems for which all sources of noise or random variations were negligible. Although such conditions would not normally give an accurate general model, there are situations in which a completely deterministic study of a communication system could give desired results.

Implementation of a deterministic Monte Carlo study would depend on the nature of the specified functions. However, since each such function is analyzed numerically, to compute outputs for given inputs in the time domain, there are few restrictions in principle on the nature of the functions. Restrictions come from the computer times required for calculation.

Modeling of memory in a system function typically leads to a differential equation representation for the function. Numerical solution of differential equations can be time-consuming computationally. Especially stringent demands on computer time result if RF, as opposed to baseband, models must be processed, or if computation must be carried through a lengthy transient period to reach steady state.

For completely deterministic inputs, a typical system with memory might require tens of minutes of large-scale computer time. Thus, almost any specified deterministic system can be analyzed in a feasible amount of computer time.

It is interesting to note that, using this method, specific time functions would be computed for the system output and each function output. The Fourier transform of any one of these time functions could be computed (using for example the fast Fourier transform) to obtain amplitude spectra.

The limited Monte Carlo approach is a term applied to a Monte Carlo approach which accounts for random inputs in a very tractable, but approximate, manner. Random inputs are accounted for by using in the simulation study only one sample function (or a very limited number of sample functions) from each random input. The single sample function is chosen to be "typical" of the ensemble of such sample functions.

Once the typical sample functions are chosen, the study is carried out in the same manner as a deterministic study. The result is a single time function for the system output and any desired intermediate outputs. By carefully choosing the typical sample functions so that, for example, its variance is close to that of the ensemble variance, certain computed outputs are good approximations of those obtained by a true statistical average. For example, using this method the spectrum of the squared magnitude of function outputs is a good approximation of the statistical power spectrum.

The limited Monte Carlo approach seems to have been used to good advantage in such problems as studying adjacent channel interference for both linear and nonlinear systems; see for example [16, 17, 18, 19, 20].

In evaluating the limited Monte Carlo approach, accuracy is a primary consideration. Unfortunately, the technique of using a single "typical" sample function from random inputs to replace what should be a statistical average is difficult to analyze for accuracy. The problem is one of statistical estimation for which confidence limits on accuracy would be desired. However, the difficulty of the statistical estimation problem is comparable to the underlying communication system problem, so that in most uses of this method accuracy is essentially unknown.

The computer time required for a limited Monte Carlo study is comparable to that used in a deterministic Monte Carlo study.

The true Monte Carlo approach removes the accuracy limitation of the limited Monte Carlo method but at the expense of requiring orders of magnitude more computer time. Typical true Monte Carlo studies are discussed in the following references: [21, 22, 23]. It is worth noting that many of the general system simulation programs use the true Monte Carlo method; see for example [24, 25, 26, 27, 28].

Unlike the limited methods, the true Monte Carlo method requires computation of an ensemble of output functions corresponding to an ensemble of input sample functions for all random inputs. Given the calculated ensemble of outputs, averaging is carried out to estimate desired parameters such as probability of error, means, variance, or power spectra.

The statistical accuracy of the Monte Carlo method is determined ultimately by how many sample functions are used in estimating the true statistical average. For many problems it is possible to compute confidence intervals for parameter estimates. In typical cases thousands of outputs must be averaged, and each output requires essentially the same computation time as one output for a deterministic or limited Monte Carlo study.

The Chernoff bound [29] can be used to estimate the number of samples required in a typical Monte Carlo study to determine probability of error. A straightforward calculation results in Table 2-2 which tabulates the number of samples required as a function of fractional error in estimating probability of error for error probabilities of 10^{-3} and 10^{-6}.

TABLE 2-2

NUMBER OF SAMPLES REQUIRED IN A MONTE CARLO STUDY

	Error Probability	
	10^{-3}	10^{-6}
Percentage Error	Number of Samples	
10%	2.9×10^4	2.9×10^7
20%	1.9×10^4	1.4×10^7
100%	2.1×10^3	2.1×10^6

This table shows that on the order of 10,000 samples must be processed to achieve reasonable accuracy for an error probability of 10^{-3}. In a typical case of a 4-phase PSK system studied with a large scale computer, approximately 0.1 second/sample was required to compute a single output. Thus 10,000 samples require on the order of 15 minutes of computer time. A similar calculation shows that 15,000 minutes of computer time are required to produce comparable accuracy for 10^{-6} error probability.

A number of methods have been reported in the literature for combining the analytical/numerical and Monte Carlo methods. Such combined Monte Carlo methods can solve practical problems in reasonable computational times, and in fact seem to offer the most potential of all the methods studied.

Unfortunately, there are two related shortcomings of the combined methods from the point of view of the objectives of the present effort, namely: they are special-purpose, and they must be tailored by a skilled analyst. On both counts they are not suited to general interactive implementations for which the equations for a specified system are organized and put together by the computer software.

Some effort during the preliminary phases of the present contract was directed toward the development of a method which has the attractive features of the combined methods, but which in addition removes the two shortcomings of these methods. The work has resulted in the development of a "modified" Monte Carlo method which is general in nature, is characterized by function equations which can be assembled in the same manner as for the true Monte Carlo methods, and has the attractive computational features of the combined Monte Carlo methods.

The chief shortcoming of the true Monte Carlo method is the excessive computer time required to compute small error probabilities. The unreasonable computer times, in turn, are caused by the fact that literally millions of correct bits must be processed through the equations for a system in order to get one bit in error. The combined Monte Carlo approach owes its success to the use of analytical methods to express the probability of error in a low-noise environment, thus avoiding the difficulty just discussed. Actually, in the combined method the analytically expressed probability can be conditioned on fixed patterns for other random inputs, such as signals, and averages with respect to these variables can be carried out with Monte Carlo methods used subsequent to the analytical determination of probabilities with respect to the noise.

The modified Monte Carlo method has been devised as a means for generalizing and systematizing the analytical computation of probabilities with respect to noise, possibly conditioned on fixed patterns, for other

variables. The method has the desirable feature that it works with the function-by-function system equations which are assembled by the computer software for use in a subsequent Monte Carlo study or for use in calculations.

Considerations involved in a modified Monte Carlo approach are illustrated by a study of a typical communication system structure such as shown in Figure 2-15. In the usual case, a complete statistical characterization of the noise inputs is given along with a similar characterization of the discrete sequence of transmitted messages. The encoder, the transmitter, the channel, and the basic receiver are described by equations (or representations) selected by the analyst. A discrete decision variable, obtained by sampling appropriate variables at the output of the basic receiver is observed by a decision function with specified regions which correspond to the possible output variables.

In operation, the complete system maps a sequence of discrete messages $\{m_n\}$ into a corresponding sequence of received messages $\{\hat{m}_n\}$. The key steps in the process can be represented by the equivalent system structure shown in Figure 2-15(b). The decision variable Y_n, at the n'th sampling instant, can be expressed as a function of the following variables:

(1) the noise inputs, $N(t)$; for all past time,

(2) the sequence of message inputs $\{m_n\}$, and

(3) the impulse response, $h(t)$, of the complete system through the basic receiver.

Mathematically this can be written as

$$Y_n = f[m_n, N(t)] \quad \text{for all i and } t \leq nT. \tag{2-25}$$

Note that $\{m_n\}$ is a sequence with known statistics, $N(t)$ is a random process with a known statistical characterization, and $h(t)$ is a deterministic time function which can be computed.

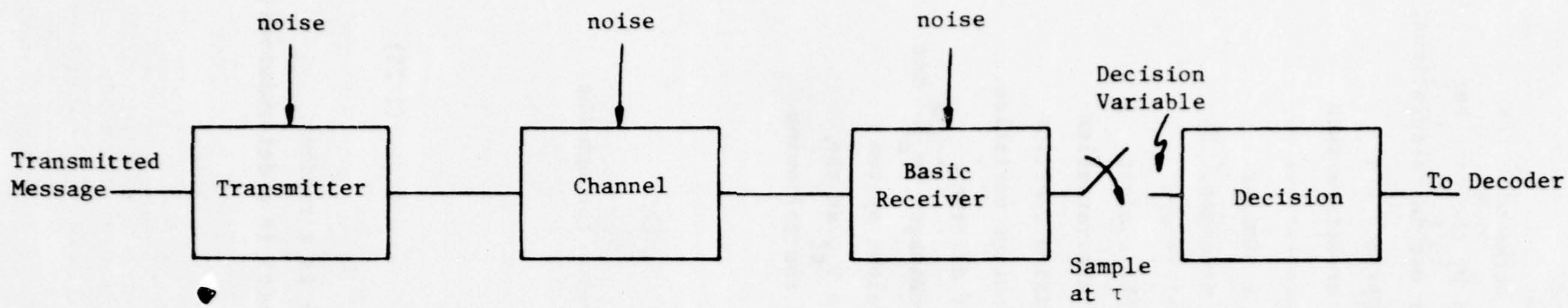


Figure 2-15(a). Typical Communication System Structure.

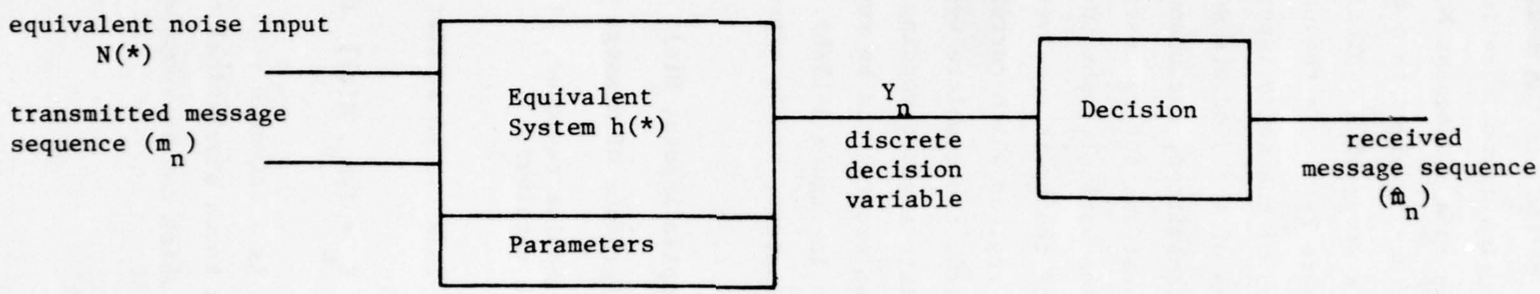


Figure 2-15(b). Equivalent System Structure.

In implementing the modified Monte Carlo approach, it is convenient to fix the message sequence so that it becomes no longer a random variable. Thus Y_n can be expressed as a deterministic function $f^*[\cdot]$ of the random noise inputs N(t) for fixed $\{m_n\}$. (Note that the effect of the deterministic functions $\{m_n\}$ and h(t) can be included in the definition of f*). Mathematically this gives the equation

$$Y_n = f^*[N(t)] \quad \text{all } t \leq nT. \tag{2-26}$$

This equation is a deterministic mapping of a random process N(t) on $(-\infty, nT)$ into a random variable Y_n. In principle, knowing the statistical properties of N(t) and of the function f*, the density function of the random variable Y_n can be computed.

The final step in the operation of the communication system is the decision process which maps the space of Y_n values into received messages (every Y_n goes into one $\hat{m}_n$). An error occurs if a given transmitted-message bit produces a value of the decision variable which maps into a value other than the correct one. This is illustrated in Figure 2-16 for a binary message and Gaussian noise.

In the context of the example illustrated in Figure 2-16, the key step in the modified Monte Carlo method is the realization of the fact that the probability that Y_n, given that m_1 is transmitted, is less than -d (so that an error occurs), is equal to the probability that the noise process N(t) has followed a trajectory which produces a value of Y_n in the error range. Mathematically this fact is given by

$$\text{Prob. Error} = P\,\{Y_n \in (-\infty,-d)\} = P\,\{N(t) \in A\} \tag{2-27}$$

where A is the set of all points such that $f^{*-1}(Y_n) \in (-\infty,-d)$.

The result illustrated by the example is general. That is to say, the probability of error, which is expressed by the probability that Y_n is contained in a certain region of its space, is also equal to the probability that the trajectory of N(t) is contained in a set which is defined through the inverse of the f* function. Once the region of the n(t) trajectory causing error is known, the probability of error can be computed by appropriate integration over the known density functions of N(t).

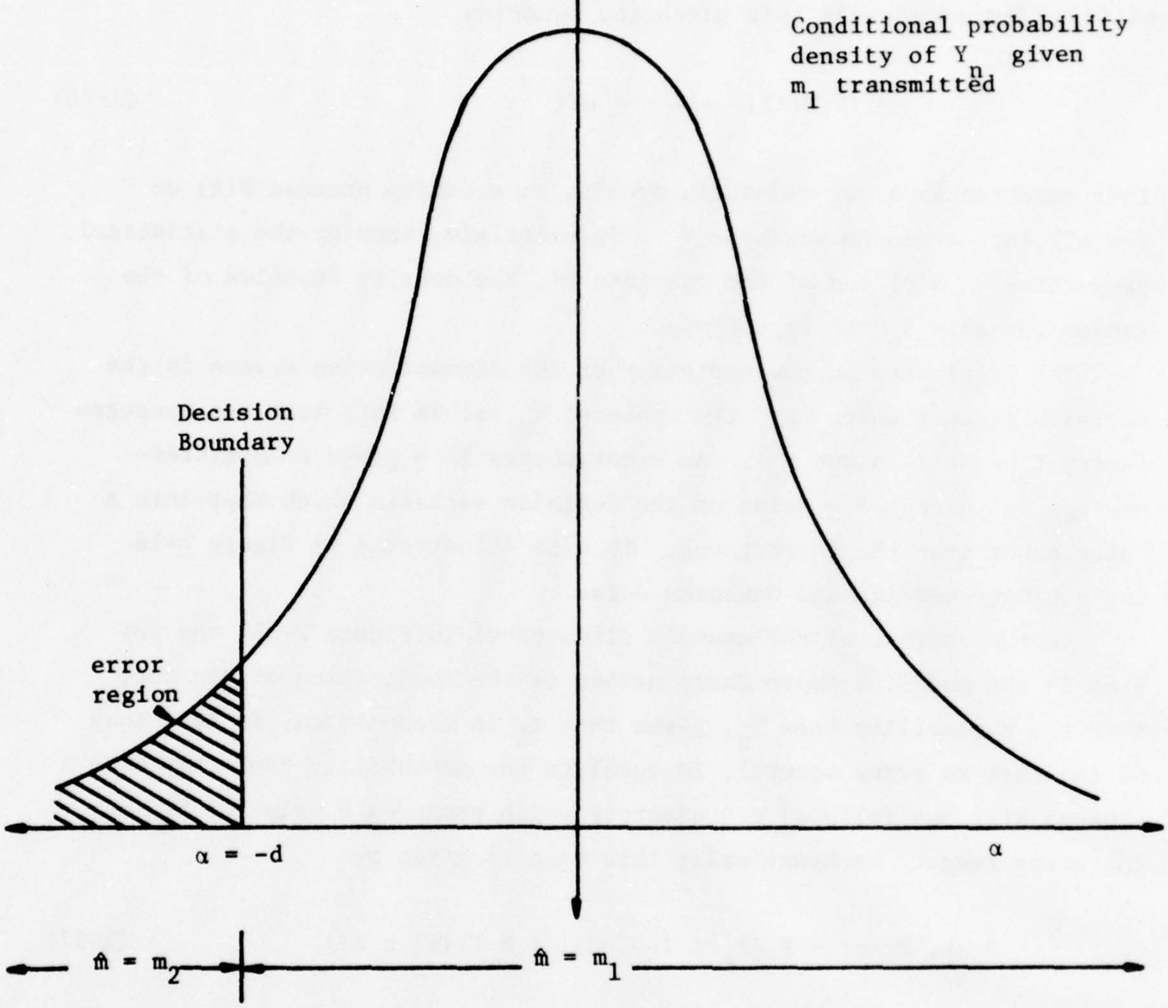


Figure 2-16. Illustration of An Error for A Binary System.

The approach of the modified Monte Carlo method is to find the error set, A, for N(t) through an efficient search procedure using decision boundaries determined from the assembled function models. Once this set is found, the error probability (which in most cases will be conditioned on certain message sequences) can be computed by appropriate integration over the given statistics of N(t).

Of the simulation techniques examined as a part of this study, the modified Monte Carlo approach seems to be the best overall, but, as in the case of the modeling approaches, this general superiority is not so overwhelming as to allow one to totally ignore the possible use of any other technique. In fact, the availability of several exercise techniques would be in harmony with the multilevel analysis stressed in the general MIDACS concept.

3. PROGRAMMING STRUCTURE FOR MIDACS

3.1 Approach to Software Development

Software is a major cost factor in the design of many computer-based application systems. For this reason it is very important to use every practical technique for reducing the software development costs without sacrificing the quality of the finished product. The development approach for MIDACS must also take into account the possibility that the designated host computer may not be available for use until some time after the MIDACS development is underway. The programming approach described below insures that facilities and resources are used efficiently, that some parts of the MIDACS system can be made operational while development continues, and that software can be transferred from one host computer to another with minimal modifications.

One technique for efficient software development is called top-down programming. The software engineer first decides on the general features of the application program, and then he constructs a model in which the general features are broken down into sets of tasks, which in turn are broken down into specific operations, and so on, until the detail is sufficient to specify each subroutine needed to fulfill the goals of the application. Then the designer uses this procedure again for each subroutine, until it reveals the actual program statements required to implement the operation. The various levels of detail in the top-down definition often provide clues for implementing a programming hierarchy, where one subroutine at one level performs a task by calling several subroutines at a lower level, each dedicated to one phase of the task. The end result of the top-down design process is a large program composed of short, identifiable, separable, logically connected subroutines.

Structured programming is another technique for efficient software development. Any large scale software project usually requires the services of several programmers working together. To insure that their efforts are coordinated and that the components of the software are compatible, a task leader identifies certain programming conventions to be followed for various

phases of the development. Examples include the names and sizes of common arrays to be used, names and functions of control variables to be set, and the number and types of arguments in the subroutine call. The details internal to the subroutines are left to the skill of the individual programmer. Once written, the subroutines are then incorporated into the application program with little or no modifications. In this fashion the structured programming method distributes the work load and speeds up the implementation process.

The MIDACS software will be developed using both top-down and structured programming techniques. Figure 3-1 shows how the combination of these two approaches results in a versatile hierarchical framework that provides for fast implementation and continuing system expansion. Notice that subroutines can be grouped into classes such as secondary controllers, subfunction simulators, etc. Programming tasks can be distributed among several programmers by defining a set of input conditions to be satisfied by each subroutine of a given class. Conditions include the names of common arrays to be accessed, the number and type of arguments in the subroutine call, and any operations that must be performed as part of general housekeeping duties. Additionally, if a class of subroutines must return information to the calling program, the nature and location of this information must be specified. In many cases, the subroutines in one class will be calling other classes of subroutines. For example, one controller will call function simulators while another calls system level analysis routines. In this case the calling programs, while belonging to the class of secondary controllers, do not necessarily have compatible outputs. That is, the input conditions for the simulators will likely be different from the input conditions for the analysis routines. The result is that in some cases the individual programmer will define a set of conditions for a new class of subroutines to be called from the program that he has developed.

Finally, notice that subroutines D and E in the figure both access the class 4 subroutines. Here the output conditions of D and E would have to be the same, since both must match the input conditions of class 4.

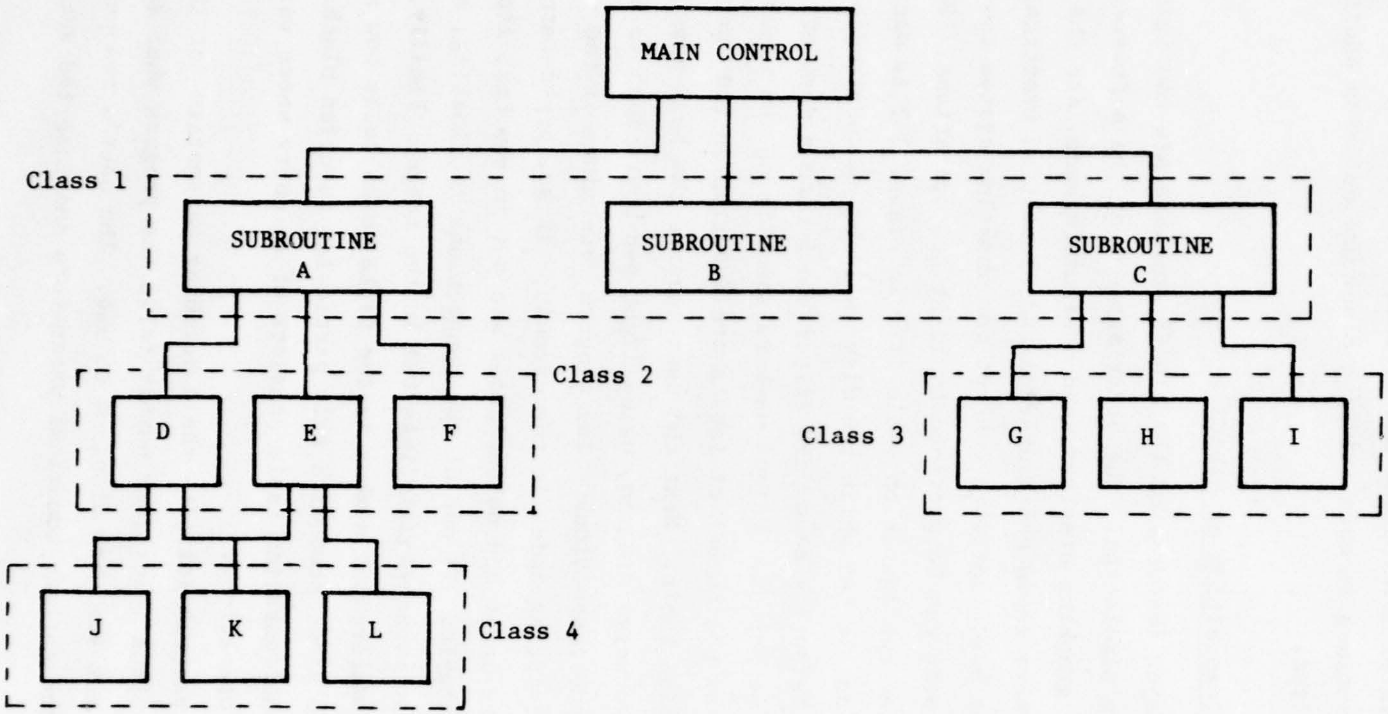


Figure 3-1. An Example of Subroutine Class Groupings and Hierarchy for Structured Programming Approach.

With this programming approach, the various classes can be set up quickly and the minimum system implemented with a handful of subroutines in each class. Then system development and expansion can continue by adding the necessary subroutines in each existing class and defining additional classes as may be required.

3.2 Software Composition of MIDACS

The two major features of the MIDACS software are configuration management and system simulation. The other aspects of the software, such as interactivity, graphics displays, and user assistance, are the "luxury options" that make rapid configuration and simulation practical. Figure 3-2 illustrates the basic software concept for managing system configuration. The control inputs can be provided by light pen operations. For instance, the data for the configuration table ITYP in Figure 3-2 is derived from the block diagram on the CRT while the diagram is being constructed. Assume that the user selects a band-pass filter block from a menu of function blocks displayed on the screen. Unique numbers identifying the block type, band-pass filter, and the number of inputs are inserted in the current line of the configuration table. Next the user inserts the band-pass filter in a partially constructed diagram, using light pen operations to establish the input and output connections. The program determines if the connections were made to existing nodes in the circuit. If so, pre-determined numbers identifying the input and output nodes are put in the next line of the configuration table. If one of the connections is floating, a new identifier for that connection is inserted in the table. Finally, the program inserts a pre-determined number in the table which tells how many describing parameters are associated with this particular function block. Thus, each line of the configuration table consists of an entry whose value depends on light pen operations.

Now the program solicits the describing parameters for the band-pass filter. The block identifier number tells the program what sequence of questions to ask and what displays to use. The user's responses are converted into the correct numerical parameters and inserted sequentially in

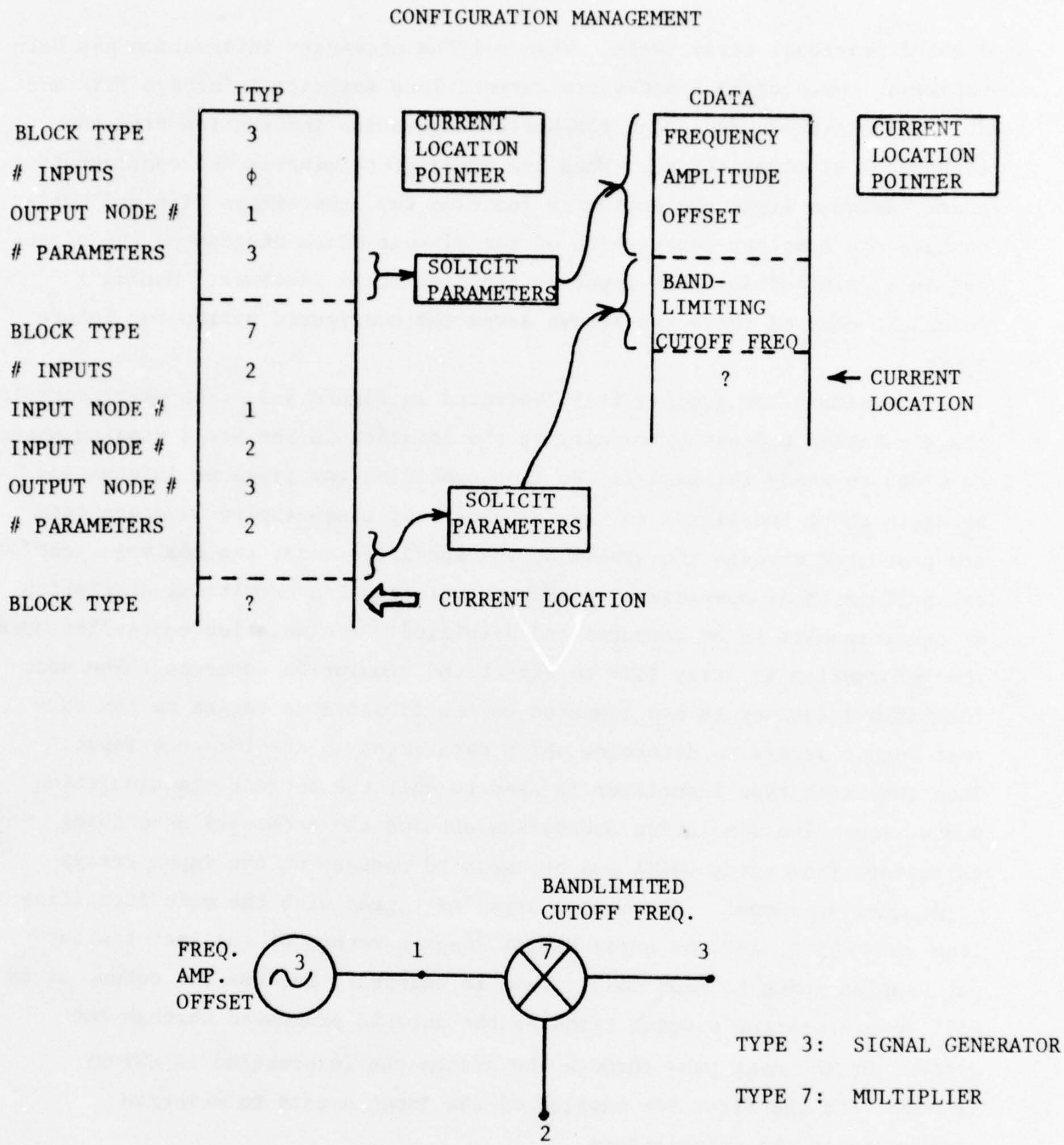


Figure 3-2. Example Showing the Numerical Description of a Block Diagram.

a one-dimensional array CDATA. When all the necessary information has been supplied, the program updates its current line pointers in arrays ITYP and CDATA and is ready to accept further configuration information from the light pen/CRT interactions. When the operator terminates the configuration phase, using a light pen action or function key, the arrays ITYP and CDATA contain the complete description of the circuit block diagram on the screen, but in a form suitable for input to the simulation routines. Making a permanent copy of these two arrays saves the configured system for future study.

The simulation process is illustrated in Figure 3-3. The user controls the simulation process by specifying the location in the block diagram where he wants to study the signal. He also specifies the types of information he wants about the signal so that as blocks of time-sampled waveform data are processed through the system to the specified node, the analysis routines can perform their operations on each output array, accumulating statistics or other results to be computed and displayed. A simulation controller uses the information in array ITYP to direct the simulation sequence. The node identifiers for inputs are compared to the identifiers tagged to the current output arrays to determine which data array to use for each input. Then the block type identifier is used to call the appropriate simulation subroutine. The simulation subroutine obtains the necessary describing parameters from array CDATA and proceeds to operate on the input arrays to produce an output. The output array is tagged with the node identifier from array ITYP, and the array HSTORY keeps a record of the last few output samples going to each node. This is necessary because the output array will be overwritten several times as the data is processed through the system. On the next pass through the system the information in HSTORY is used with the first few samples of the input arrays to maintain continuity in the calculations.

3.3 Programming Languages and Operating Systems for MIDACS

Although the demonstration program discussed in Chapter 7 is written completely in FORTRAN IV, some parts of it could have been implemented more

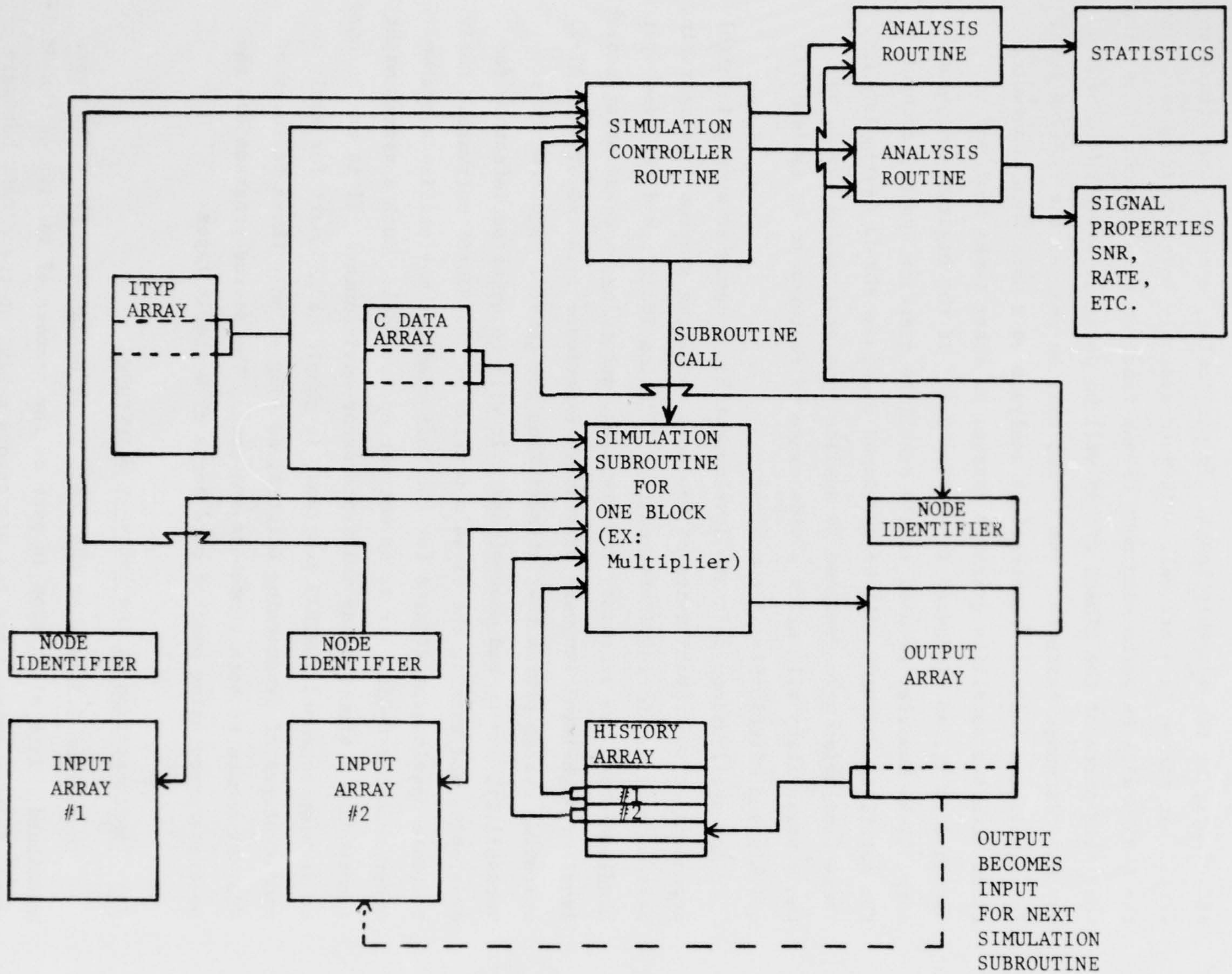


Figure 3-3. Diagram of Simulation Process Showing Data Paths.

efficiently at the assembly level. In particular, encoding operations on serial bit stream data are well suited to assembly level approaches. Since the C-8500 will be performing most of the function simulations, it is likely that some parts of the MIDACS system will be designed using the C-8500's assembly language features. The C-8500 will be used at the FORTRAN level for simulation control, mathematical analyses on a data base, communications with the satellite graphics system, or other tasks that may be assigned to it, as discussed in Section 3.4. If the Digital Equipment Corp. GT-44 computer is used as the satellite graphics system, much of the satellite software will be developed using the PDP-11 System Library. These facilities were designed as applications programming tools, and their versatility will allow a wide range of features to be implemented quickly and efficiently in the GT-44.

Any applications software development for a computer system is highly dependent on the capabilities of the host operating system. The operating system contains the compilers, editors, linking loader, and run time system handlers which are necessary for creating, debugging, and executing assembly level or high-level language programs. Experience with the GT-44's RT-11 operating system proves that it satisfies the general requirements of versatility, power, and ease-of-use. It will be quite satisfactory for developing and running the GT-44's portion of the MIDACS software. Early software development efforts for the host computer may utilize a system other than the C-8500, if it is not yet operational. Such a system might incorporate a time-sharing multi-processor environment. If it can be used to develop standard FORTRAN programs, it should be adequate for early to middle stages of programming efforts, as long as only those features of FORTRAN common to both computers are used. The source programs can then be transferred using punched card decks or magnetic tapes.

3.4 Implementation of the Software Structure

In Section 3.3 some of the tasks to be assigned to the C-8500 were mentioned. Task allocation is part of the process of setting up the division of labor between the two processors making up the MIDACS hardware

configuration (Figure 3-4). This configuration exploits the advantages of both computer systems while minimizing their disadvantages. The graphics capabilities of the satellite provide for fast responses to the user while the speed and memory of the host make it suitable for the massive numerical computations and manipulations of large data bases required for MIDACS. Using two computers presents problems involving the additional hardware and software requirements, installation and maintenance, and dealings with the manufacturers; but the benefits to be gained justify the extra effort.

A programmable satellite system like the GT-44 can easily handle all of the lexical, syntactical, and semantic processing involved in maintaining high interactivity with the user. All of the graphics functions and operations can take place in the satellite, thus reducing considerably the amount of data sent over the communications link. Now the problem arises of how to divide the application program between the two computers since the satellite is obviously capable of handling a large share of the operations. Also, both computers may at times require access to the same data. For example, the host may use a data array DATA to get sets of parameters during a simulation. The satellite must access the same array for configuration and modification. If separate copies of the data are kept in each computer's memory space, the program must insure that the arrays are synchronized, that is, pointers and values must be kept up to date in each computer as one or the other accesses or changes entries in the array. Data accessed frequently by one processor and infrequently by the other might best be kept in the one accessing it frequently. Then the communications link would provide the access path for the other processor. The approach taken for any one case will take into account the amount of data involved, how often each computer accesses the data, and the effect on response time of the different approaches.

The subroutine classes shown in Figure 3-1 make no distinction between the two processors since some classes may be implemented in the host, some in the satellite, and some perhaps split between the two. A communications protocol would be set up to handle the transfers of program flow between the subroutines in each processor. For example, a configuration controller

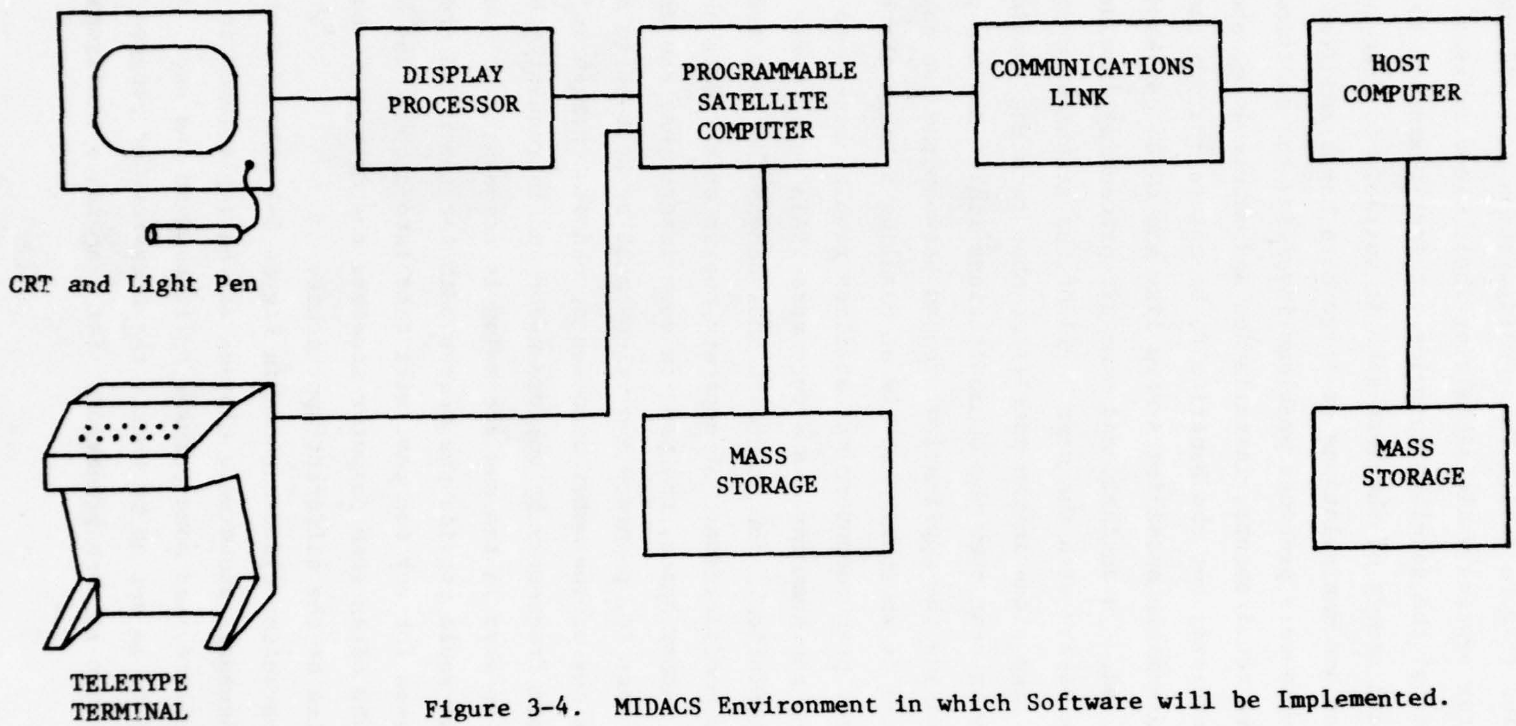


Figure 3-4. MIDACS Environment in which Software will be Implemented.

and a set of graphics routines are resident in the satellite and allow the configuration of a communications system from a set of pre-defined elements. A certain signal from the user is recognized by the controller, it transfers data describing the configuration to the host, and informs the main control program to call the simulation controller in the host, whereupon the system whose diagram is on the CRT is modeled and analyzed in the host.

The job of redistributing processing tasks between computers, should it be deemed worthwhile, would not be impossible; but it could become somewhat involved. Brown University and the University of North Carolina have developed satellite-host graphics systems that allow a programmer to practically ignore the fact that there are two computers when writing programs. Each module or subroutine can be assigned at will to the host or satellite, and the assignments can be changed without reprogramming. The interprocessor communications are taken care of by the run-time handlers in the operating system [30]. Unfortunately, this facility is not available in the intended computer system for MIDACS. However, the degree of program shuffling required should not be great once a pattern for distributing the tasks is laid out. Basically, the satellite would handle all interactions with the user and create the resulting set of data to be sent to the host. It would also receive from the host data output by analysis or simulation routines, to be converted into graphical displays. The host would be concerned primarily with the time-domain processing, the Monte Carlo evaluations, and other large scale computational tasks performed on its resident data base and influenced by variables received from the satellite.

The schedule for implementing the MIDACS system can be outlined in general terms. First, of course, the hardware facilities must be operational. This means that a host computer, a satellite graphics system, and a communications link are configured together and the respective operating systems are shown to be functioning properly. Keep in mind that the host will not necessarily be the C-8500 during the early phases of the effort. Next, a communications protocol is set up to effect transfer of program flow between computers. This can be as simple as using FORTRAN read and write statements that address the communications link as a logical unit

number, and designing the software so that one processor is waiting for data from the link (READ) when the other starts transmitting (WRITE). The main controller can then be installed in the host computer and the configuration controller and associated graphics routines installed in the satellite. These programs should be tested thoroughly before proceeding to install the simulation controller in the host along with enough subfunction simulation routines to test its operation. Next a few analysis routines are added in the host and the corresponding graphical display routines incorporated in the satellite. The working software now comprises a "minimum system". Further development continues by adding function simulators, other analysis routines, and their associated graphics and I/O routines. This schedule provides a methodical approach for installing and debugging the MIDACS foundation software, and it allows the designers to implement an initial working system while continuing the software development. Over the period of time devoted to the effort, the initial system matures into a versatile, analytical design tool.

4. SYSTEM LIBRARY

4.1 General Element Characteristics

The elements of the system library are mathematical models of complete communication systems. Each model is capable of evaluating one or more of the performance measures identified with the particular system of interest, once the characterizing parameters have been assigned numerical values. Typically, such models would be capable of evaluating measures such as probability of bit error, signal-to-noise ratio, and signal bandwidth; however, a more complete list of both system parameters and performance measures is provided in Table 4-1. Table 4-1 should be viewed as an indication of the broad range of possibilities for specification and evaluation, rather than as a limited directory of parameters and measures to which the simulation approach is applicable.

It is important to note that many of the system models do not directly account for channel characteristics. For example, the signal attenuation caused by the channel is often not computed directly, but it is reflected in the signal-to-noise ratio which is frequently a parameter in the mathematical expressions for probability of bit error. This situation is not a serious limitation of the system level analysis, since it is possible to combine a program which analyzes a given channel with a program which evaluates a performance measure influenced by channel characteristics.

The attractiveness of the combined approach is enhanced by the recognition that channel modeling has received a great deal of attention, and several working models are readily available for use in a simulation program [31, page 33-4], [32, page I-M], [33], [34], [35], [36]. It should be recognized however that programs of this type tend either to be large in size, slow in computation, and highly accurate, or moderate in size, fast in computation, and of limited accuracy.

TABLE 4-1

Typical System Parameters and Performance Measures

SYSTEM PARAMETERS	SYSTEM PERFORMANCE MEASURES
Data Rate	
Type of Encoding	Waveform Rise Time
Block Length	Pulse Response
Constraint Length	Percent Overshoot
Type of Modulation	Waveform Settling Time
Levels of Modulation	Signal Bandwidth
Transmitter Output Power	Signal Spectrum
Transmitting Antenna Type	Signal-to-Noise Ratio
Type of Propagation Path	Probability of Bit Error
Length of Propagation Path	Noise Histogram
Frequency of Transmission	Decision Variable Histogram
Time of Day	Intersymbol Interference
Time of Year	Ratio
Receiving Antenna Type	
Receiver Noise Figure	
Type of Demodulator	
Type of Decoding	
Type of Decision Device	
Receiver Bandwidth	

4.2 Library Elements

A large variety of communication systems are currently employed by the Air Force, and it is anticipated that future technological developments will increase both the number and the complexity of the systems in service. This diversity in form manifests itself in several major areas: the type of encoding/decoding, the type of modulation/demodulation, the type of channel, and the frequency of operation. Several of the more important options for these features are listed in Table 4-2 [37], [38], [39].

The options identified in Table 4-2 were observed to be important operational characteristics of Air Force communication systems [37], [38], and therefore it is appropriate that they appear in the system level models of the simulation program. Block diagrams which are representative of the system models that should be included in the system library are presented in Figures 4-1 through 4-5. In each case, the basic system has been identified by the channel employed, since this is consistent with common Air Force practice. It should also be noted that specific forms have not been indicated for elements such as encoders and modulators. This has been done to demonstrate the flexibility of the representational form. For example, one satellite repeater system may employ FSK modulators/demodulators, and another system may employ PSK modulators/demodulators, but both systems can be described by the structure depicted in Figure 4-1.

For a given channel, any appropriate form of coding, modulation, demodulation, and decoding may be used, and each unique combination will require a different system level library element. Various combinations of channel type and operating frequency are also possible, but only certain combinations find practical application. These combinations are shown in Table 4-3.

TABLE 4-2

Features and Options for Communication Systems

CODING	MODULATION	CHANNEL	FREQUENCY
Convolutional	FSK	Satellite Repeater	Baseband
Block	PSK	Line-of-Sight	HF
	DPSK	Troposcatter	VHF
	QPSK	Wire-Line	UHF
	DQPSK	HF	Microwave

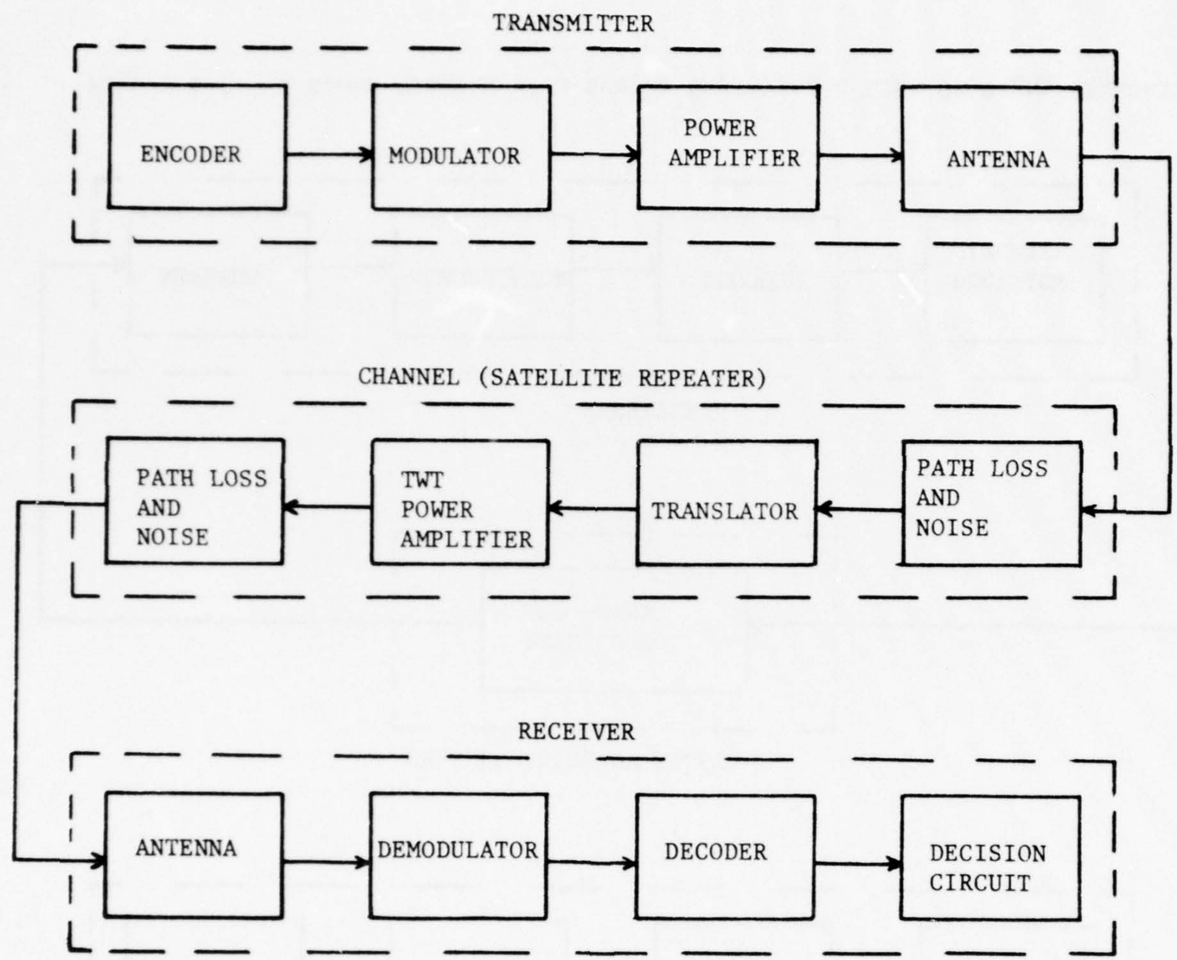


Figure 4-1. A Block Diagram of a System Employing a Satellite Repeater.

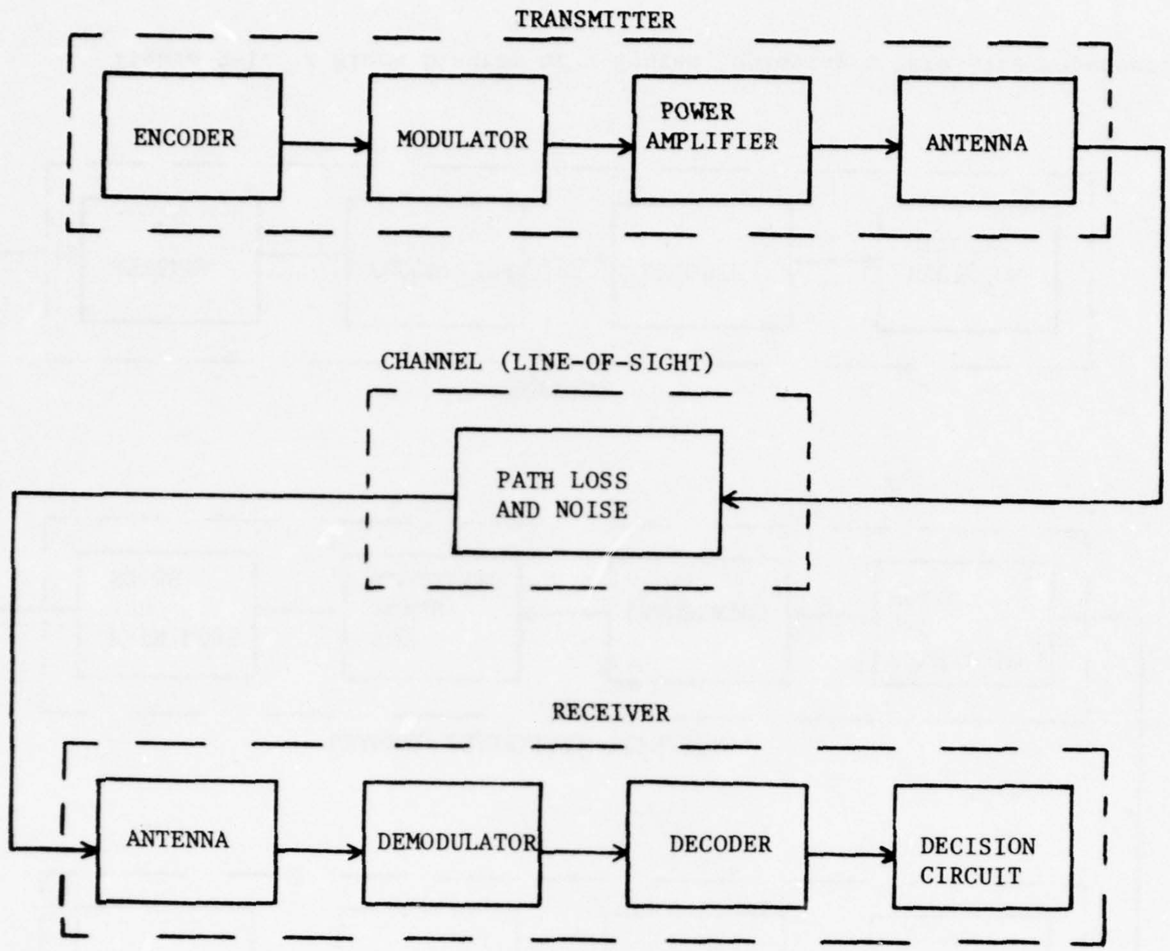


Figure 4-2. A Block Diagram of a System Employing a Line-Of-Sight Channel.

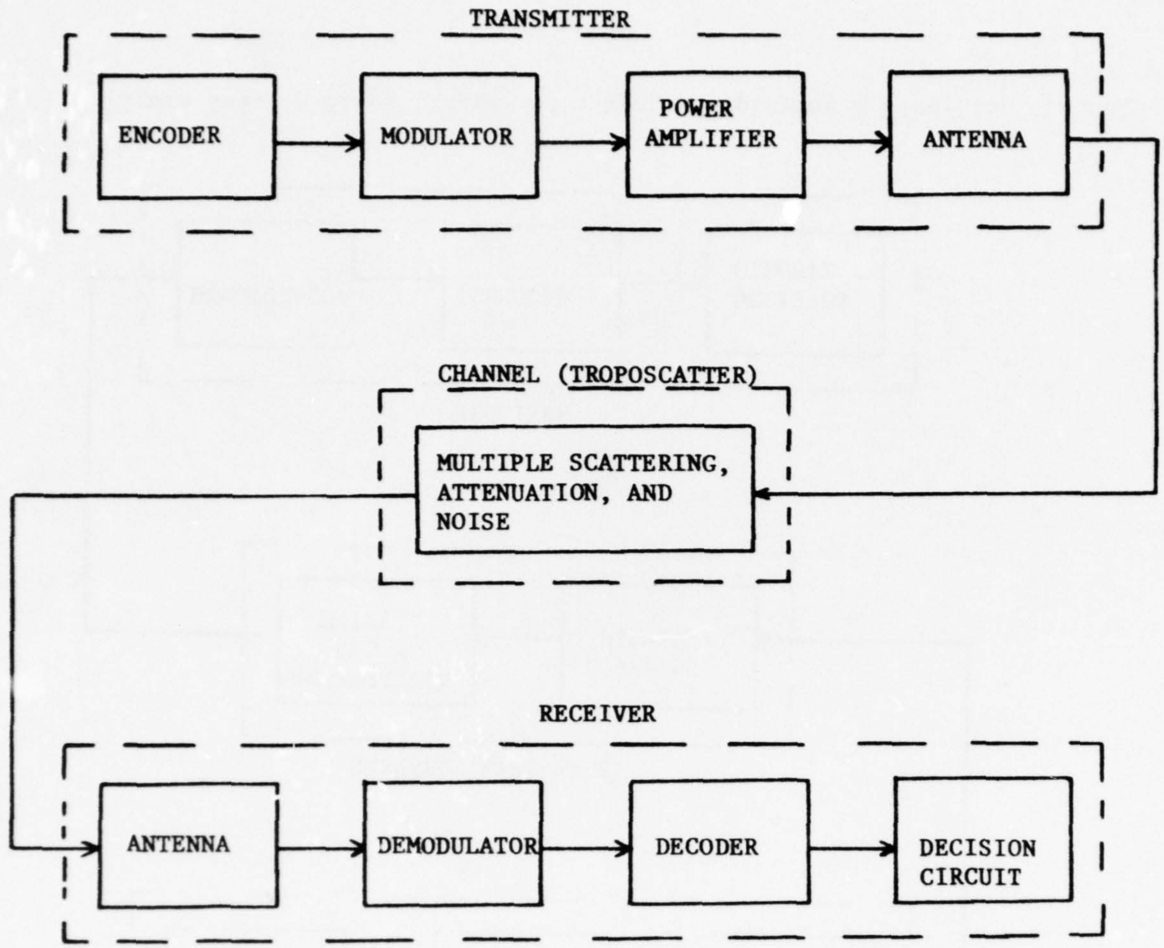


Figure 4-3. A Block Diagram of a System Employing a Troposcatter Channel.

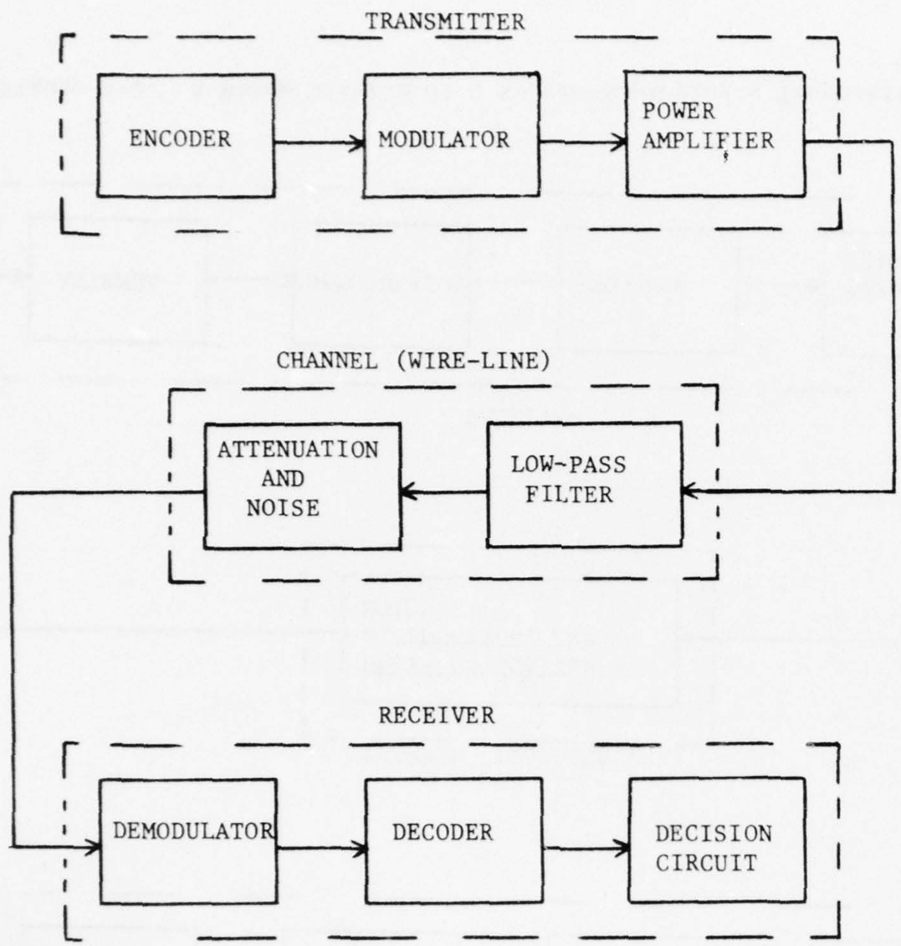


Figure 4-4. A Block Diagram of a System Employing a Wire-Line Channel.

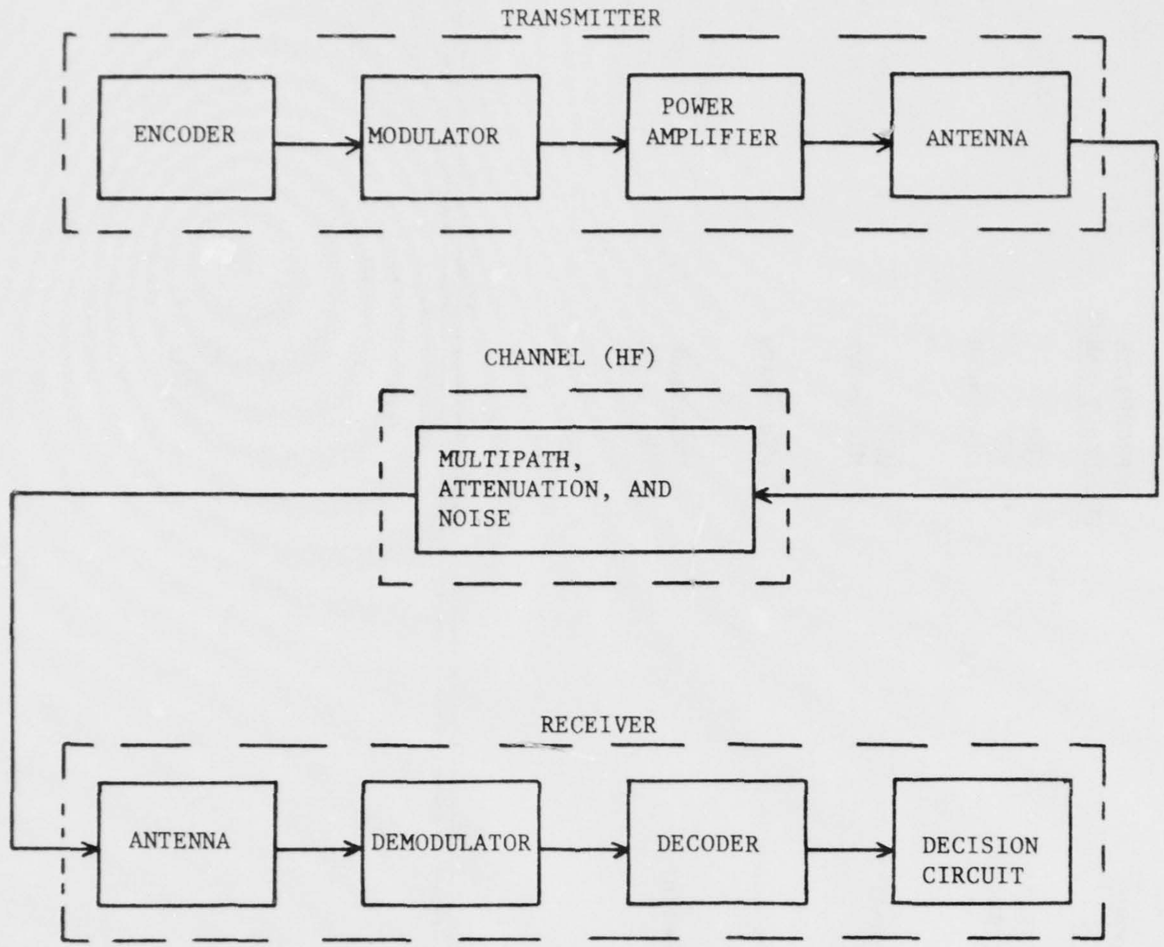


Figure 4-5. A Block Diagram of a System Employing An HF Channel.

TABLE 4-3

Channel and Frequency Combinations

CHANNEL TYPE	APPROPRIATE FREQUENCY RANGE
Satellite Repeater	VHF UHF Microwave
Line-of-Sight	UHF Microwave
Troposcatter	Microwave
Wireline	Baseband HF
HF	HF

5. FUNCTION LIBRARY

5.1 General Element Characteristics

The elements of the function library are mathematical models of individual functions which are used in structuring block diagrams of digital communication systems. Typical of the elements being considered would be a low-pass filter model. The use of such elements to describe a system's structure is demonstrated in Figure 5-1. As discussed in Section 2, these models describe the time domain relationship between the function's input(s) and output(s), and it is this common input/output form that allows the models to be connected serially such that the output of one functional block becomes the input of another block in the system's structure.

One important difference between system and function level models should be immediately apparent. The system level models represent the combined effects of many system functions, and, therefore, it is necessary to provide for the identification of the important features and parameters if accuracy is to be maintained in the modeling effort. In contrast, the function models generally describe one very distinct function, and thus a relatively small number of features or parameters is associated with any one model element. It should be remembered, however, that a realistic system model will require many functional elements, with the result being that a greater number of features and parameters must be specified at the function level than at the system level to achieve the same system model.

5.2 Library Elements

A study of the block diagram structure of digital communication systems has been made for the purpose of identifying those functions which are used to model communication systems. As a result of the study, a large number of functions have been identified, and the identified functions have been grouped into six categories: Algebraic Operations, Trigonometric Operations, Functional Operators, Logical Operations, Active

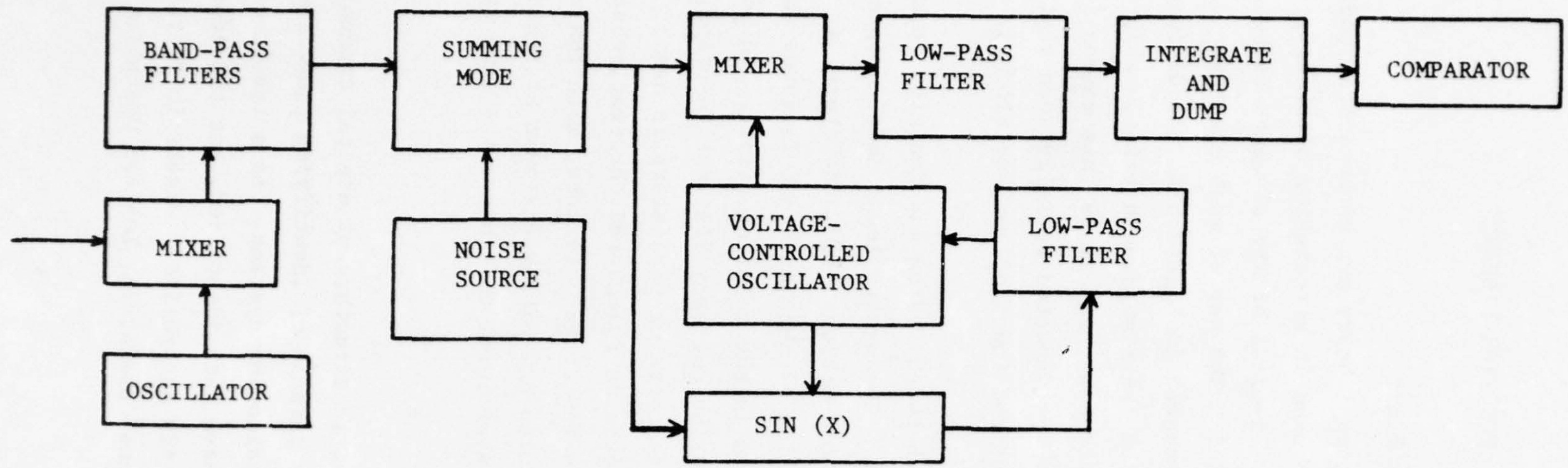


Figure 5-1. A Block Diagram Constructed with Functional Elements.

Network Functions, and Passive Network Functions. Notice that each of these categories is represented in Figure 5-1. Algebraic Operations are represented by the summing node and the mixer, a time-domain multiplication device. The phase detector, SIN(X), performs a trigonometric operation; and the integrator is a functional operator. The comparator is a part of the system's decision circuitry and performs a logical operation, while the band-pass and low-pass filters are passive network functions. Both the deterministic and random types of active network functions are a part of Figure 5-1. The oscillator is a deterministic source, while the noise source is random in nature.

One additional modeling category is necessary if a complete simulation is to occur: a channel model. As indicated in Table 4.2, a variety of channels are encountered in normal Air Force communication systems; and, therefore, it is desirable to have function level models for each of the important channel types. One might initially assume that the models employed for analyzing channels at the system level would be usable for the function-level analysis. In general this is not true since many of the channel models which are suitable for a system-level analysis calculate only average behavior, such as average attenuation. The function-level analysis requires that all evaluations be of an instantaneous nature in the time domain. As a minimum, the function-level channel model should be able to account for signal attenuation, noise contributions, signal dispersion, and multipath effects. The recommended members of the six categories and certain related information about the identified functions are presented in Tables 5-1, 5-2, 5-3, 5-4, and 5-5.

Table 5-1

ALGEBRAIC, TRIGONOMETRIC, AND FUNCTIONAL OPERATIONS

CATEGORY	FUNCTION	NUMBER OF INPUTS	NUMBER OF CUTPUTS
ALGEBRAIC OPERATIONS	Addition	2	1
	Subtraction	2	1
	Multiplication	2	1
	Division	2	1
TRIGONOMETRIC OPERATIONS	Sine(X)	1	1
	Cosine(X)	1	1
	Tangent(X)	1	1
	Arcsine(X)	1	1
	Arccosine(X)	1	1
	Arctangent(X)	1	1
FUNCTIONAL OPERATORS	Differentiator	1	1
	Integrator	1	1
	Correlator	1	1

Table 5-2

LOGICAL OPERATIONS

CATEGORY	FUNCTION	NUMBER OF INPUTS	NUMBER OF OUTPUTS
LOGICAL OPERATIONS	AND	N	1
	OR	N	1
	NAND	N	1
	NOR	N	1
	Comparator	2	1
	Mod (N) Addition	2	1
	Mod (N) Subtraction	2	1
	Mod (N) Multiplication	2	1
	Mod (N) Division	2	1
	Flip-Flop	1	2
	Shift Register	1	N

Table 5-3

ACTIVE AND PASSIVE NETWORK FUNCTIONS

CATEGORY	FUNCTION	TYPICAL PARAMETERS
ACTIVE NETWORK FUNCTIONS	Sinewave Generator	Peak Amplitude, Frequency, Phase, D.C. Offset
	Squarewave Generator	Amplitude, Period, D.C. Offset
	Ramp Generator	Amplitude, Period, D.C. Offset
	Pulse Generator	Amplitude, Duration, D.C. Offset
	Gaussian Noise Generator	Mean, Variance
	Uniform Noise Generator	Range
	Rayleigh Noise Generator	Mean, Variance
PASSIVE NETWORK FUNCTIONS	Phase Shifter	Phase Shift
	Low-pass Filter	Type, Order, Cutoff Frequency
	Band-pass Filter	Type, Order, Center Frequency, Bandwidth
	High-pass Filter	Type, Order, Corner Frequency
	Notch Filter	Type, Order, Center Frequency, Notch Depth
	Voltage Controlled Oscillator	Output Amplitude, Voltage-to-Frequency Conversion Constant

Table 5-4

PASSIVE NETWORK FUNCTIONS

CATEGORY	FUNCTION	TYPICAL PARAMETERS
PASSIVE NETWORK FUNCTIONS	n-th Law Device	Order of Non-Linearity
	Time Delay	Delay
	Hard Limiter	None
	Soft Limiter	Slope, Breakpoints
	Phase Detector	None
	Frequency Multiplier	Order of Frequency Multiplication
	Time Multiplexer	Number of Inputs, Sampling Rate
	Frequency Multiplexer	Output Frequencies
	Amplitude Modulator	Output Power, Number of States
	Frequency Modulator	Output Power, Number of States, Deviation Constant
	Phase Modulator	Output Power, Number of States, Deviation Constant
	Differential Phase Modulator	Output Power, Deviation Constant
	Coherent AM Detector	None

Table 5-5

PASSIVE NETWORK FUNCTIONS

CATEGORY	FUNCTION	TYPICAL PARAMETERS
PASSIVE NETWORK FUNCTIONS	Coherent FM Detector	Number of Tones, Tone Frequencies
	Non-Coherent FM Detector	Number of Tones, Tone Frequencies
	Coherent PM Detector	Number of Phase States
	Differential PM Detector	Bit Interval
	Sampler	Sampling Rate, Dynamic Range
	Hold Device	Order of Hold, Sampling Rate
	Quantizer	Number of Levels, Dynamic Range
	Compressor	Regions of Compression, Amount of Compression
	Expander	Regions of Expansion, Amount of Expansion
	Block Encoder	Type of Code, Block Length, Code Rate
	Convolution Encoder	Constraint Length, Code Rate
	Block Decoder	Type of Code, Block Length, Code Rate
	Convolution Decoder	Constraint Length, Code Rate
	Variable Gain Amplifier	Gain Sensitivity

6. HARDWARE REQUIREMENTS

6.1 Major Hardware Items

The major hardware components envisioned for the implementation of MIDACS are presented in Figure 6-1. They include a graphics terminal, an interface unit, and a host computer. Present plans call for a Digital Equipment Corporation GT-44 to be used as the graphics terminal and a Collins C8500 to serve as the host computer. The details of the interface unit for the identified terminal-computer combination are provided in Figures 6-2 and 6-3. The basis for the indicated structures is material prepared by RADC [40, page 11] and the equipment manufacturers [41, page 6, 21], [42, page 2].

An alternate configuration of the host computer and the graphics terminal has been identified. In this configuration the host computer operates under DOSS and the peripheral interfaces are connected to the host's record channel. The claimed advantage for this arrangement is that it provides FORTRAN support and that the data transfer rate between the host computer and the graphics terminal will be higher than that achievable with the multiplex loop arrangement. It should be noted that the latter advantage will not significantly enhance the speed of program execution.

6.2 Hardware Characteristics

The hardware interconnections indicated in Figures 6-2 and 6-3 are for the specific devices identified above and do little toward identifying the necessary or desirable operational characteristics achieved by the particular hardware elements. Those characteristics will now be discussed for each of the three major hardware components.

The interactive graphics aspect of MIDACS is central to the overall simulation approach, and therefore, the operational characteristics of the graphics terminal are of great importance. For the simulation effort to be carried out in a manner consistent with the current MIDACS concept, it is necessary that the graphics terminal employed have the following features: 1. A light pen, 2. A graphics output capability, 3. A graphics input capability, and 4. A communications handler for conversing with the host computer. Features which are desirable but not considered to be absolutely necessary are:

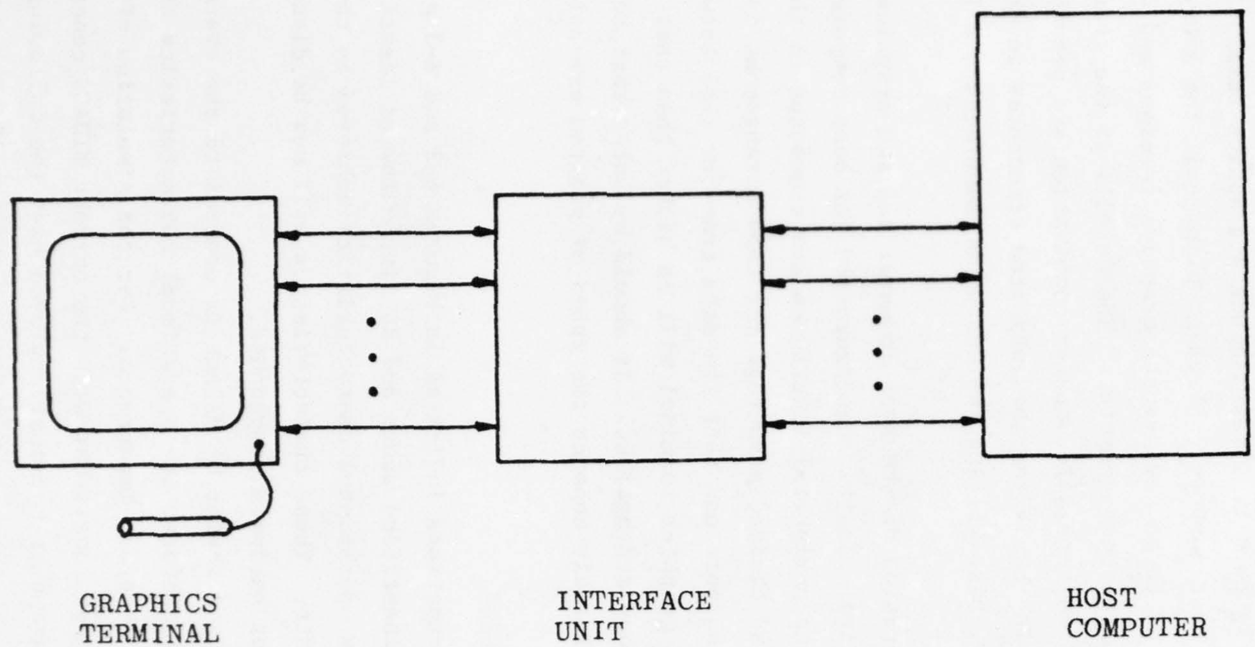


Figure 6-1. The Major Hardware Components of MIDACS.

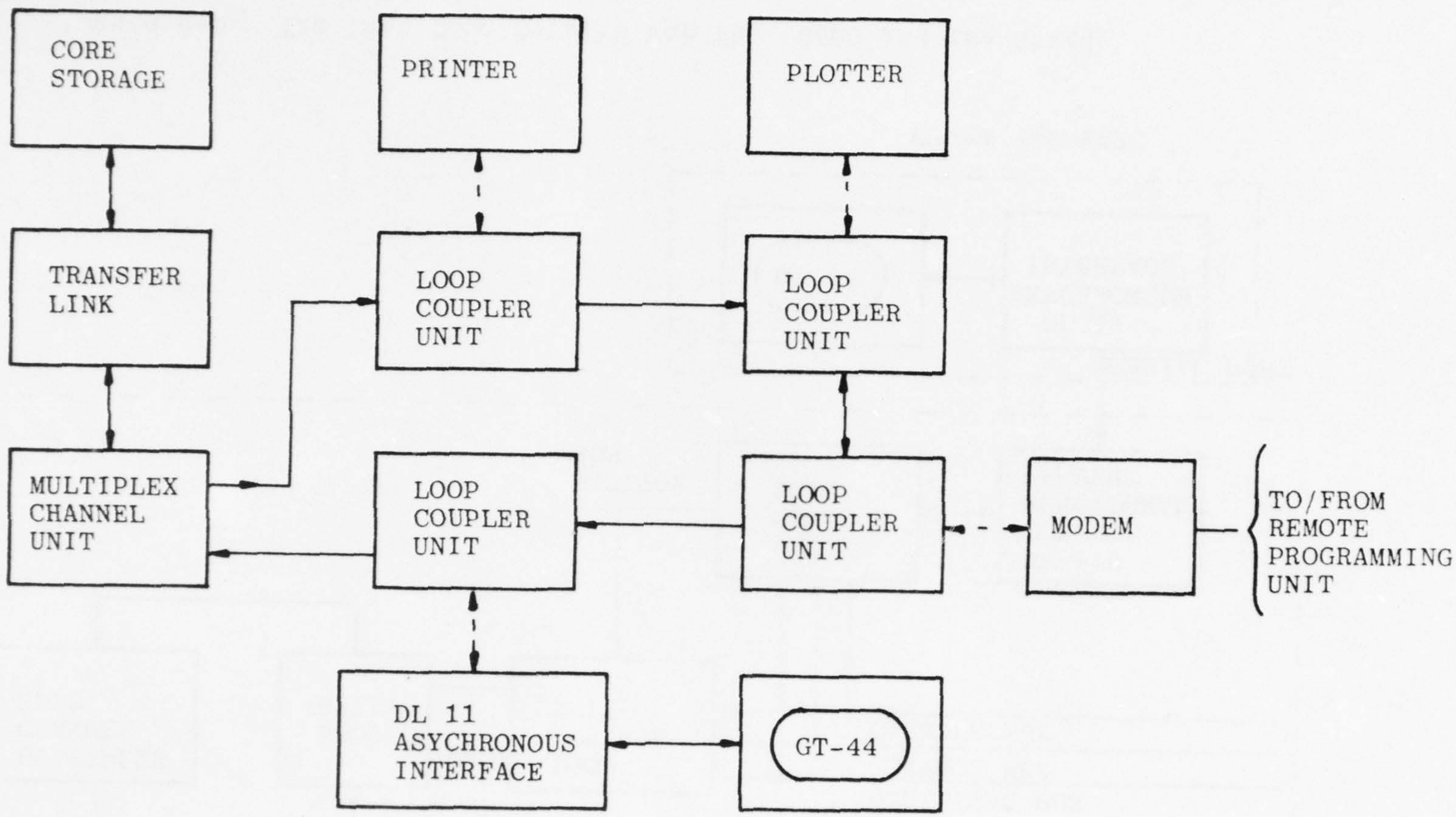


Figure 6-2. The C8500 Interface Scheme.

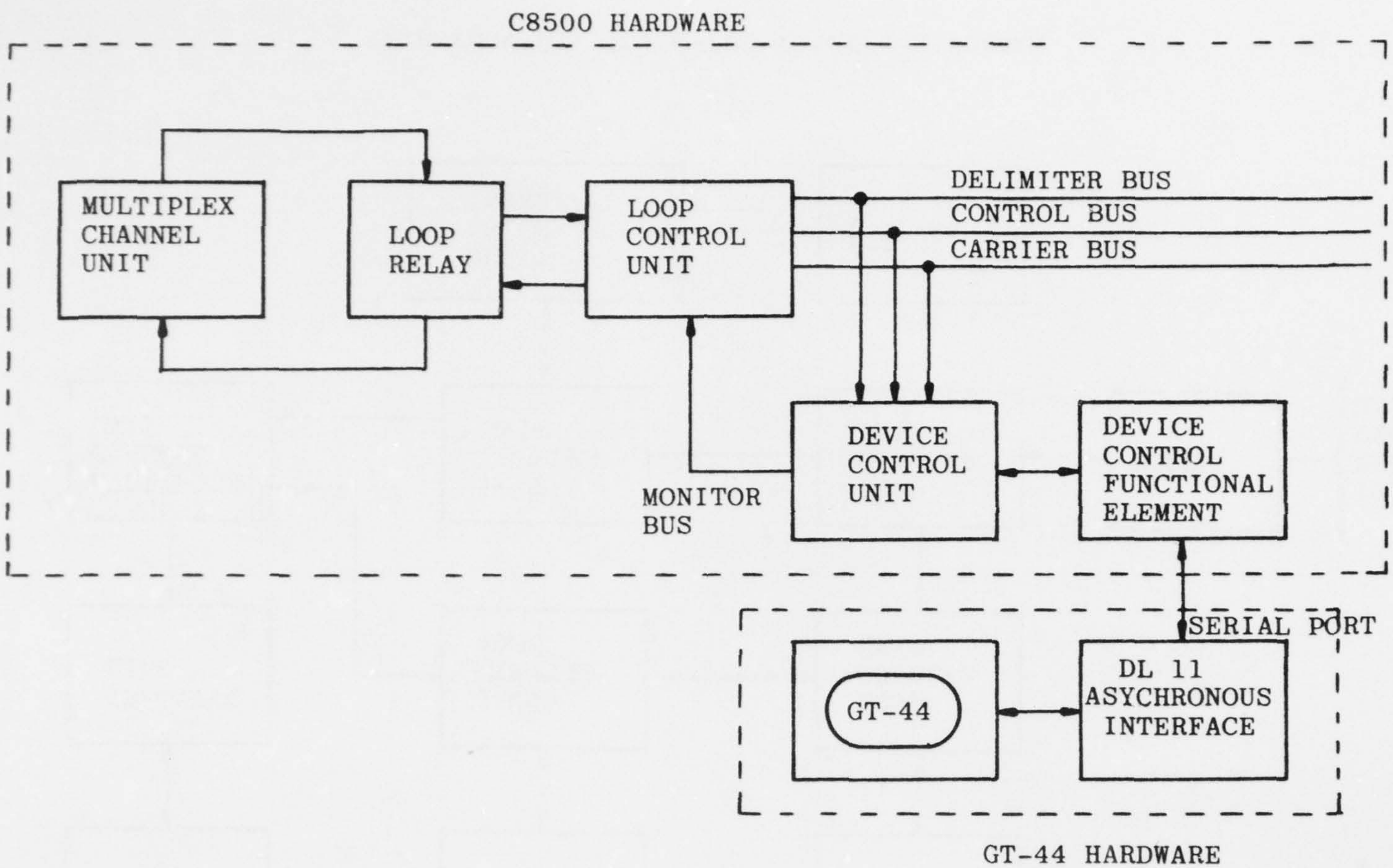
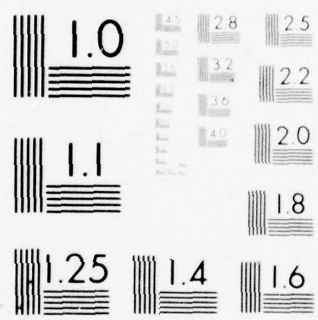


Figure 6-3. The Interface Details For The C8500 and the GT-44.

AD-A037 833 GEORGIA INST OF TECH ATLANTA ELECTRONICS TECHNOLOGY LAB F/G 9/2
INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY.(U)
FEB 77 R W MOSS, R W RICE, D R SENTZ F30602-75-C-0247
UNCLASSIFIED RADC-TR-77-42 NL

2 OF 4
AD A037833



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

1. Local memory,

2. Local mass storage,

3. A resident graphics support package, and

4. System software for an intelligent terminal mode of operation.

Consider those features indicated above as being necessary for the graphics terminal. The light pen is the means by which the program user will interact with the simulation software. Questions posed by the control program will be answered and exercise options will be indicated by pointing a light pen to the appropriate screen location. As such, the light pen is a part of the graphical input capability of the program, but its potential in this area may be greatly increased by using graphical number line techniques so that quantitative data may be entered using the light pen.

In the present context, graphical output is interpreted to include both the structural block diagrams generated in the process of modeling a system and the graphs of data generated in the analysis of those systems. As described in Section 2, the block diagram has been chosen as the means by which the analyst will define the structure of the system he wishes to study. It is important, therefore, for a means to be available to display the block diagram of the system, and the graphics terminal has been chosen as the display medium because of its speed and flexibility.

There are two basic forms which the graphics terminal can assume. It can be a slave terminal to the host computer and have no processing capabilities of its own, or it can be an intelligent terminal capable of expediting the overall simulation effort. In either case it is mandatory that a communications handler for processing data to be displayed by or input through the graphics terminal be available.

The four features identified as being desirable are oriented toward making the graphics terminal an intelligent terminal. This is deemed to be desirable since it would result in a more efficient allocation of computational resources than is possible when the graphics terminal is a

slave unit. The use of an intelligent terminal will allow the generation of graphical display information and numerical computation to occur in parallel, and thus the overall processing time can be reduced when compared to the time required for processing in a strictly serial form. The user will perceive the difference as a higher degree of interactivity for the system using the intelligent terminal.

The second important hardware item is the host computer, and in order to implement the simulation program as currently envisioned, the host computer should possess the following:

1. A system software package capable of efficiently operating the host computer,

2. A FORTRAN compiler complete with an extensive library of mathematical functions,

3. A graphics support package designed to operate in the host computer's environment and drive the associated graphics terminal,

4. Sufficient memory size to allow the handling of moderately complex computations efficiently,

5. Sufficient mass storage to allow the accumulation of sizable data arrays and the retention of a large number of graphical routines, and

6. A high speed central processing unit.

It is considered neither necessary nor desirable that the software development phase of MIDACS include the development of the basic system software for the host computer. This is in recognition of the fact that most computer manufacturers have expended a great deal of time and money to develop such software packages and that comparable expenditures would be needed on the part of an independent developer to produce an alternate package. This principle does not preclude the limited modification of an existing software package for the purpose of optimizing computer performance in the simulation environment.

There are certain advantages in using a standard language such as FORTRAN, but as was pointed out in Section 3, this restriction would cause certain models to be very inefficient in program size and execution time. Nevertheless, if the programming is to be carried out in FORTRAN, it is necessary that the host computer have a FORTRAN compiler, and furthermore, since the simulation program will perform many complicated computations, it is desirable that the FORTRAN version employed contain an extensive library of mathematical functions. A list of the more important members of the mathematical library is provided in Table 6-1.

The association of the graphics support package with the host computer rather than with the graphics terminal indicates the possibility of using a slave terminal, but this is not the recommended approach. On the grounds of processing speed and flexibility, the use of an intelligent terminal which handles all graphical interactions is recommended. In either case, a graphics support package is necessary, and the package should allow the development of graphics routines by commands compatible with the FORTRAN language [43].

The amount of memory space required to accommodate a large simulation program is difficult to approximate in advance; however, the development of the demonstration program described in Section 7 has provided a sound basis for saying that significantly more than 16K of memory would be necessary. Current estimates indicate that the 64K memory available in the C8500 will be adequate for the development and operation of a simulation program of moderate size; however, based upon the experience gained in developing the demonstration program, the 16K memory currently available in the GT-44 owned by DICEF is inadequate for extensive graphics development. This latter observation pertains to the use of the GT-44 as an intelligent terminal.

The availability of mass storage is important since it is anticipated that many of the graphical displays used in a large scale simulation program would be generated during program development and would occupy memory space only when specifically called upon by the user. Also, it has been pointed out in Section 2 that the time domain analysis will involve the generation of large data arrays which are most efficiently

Table 6-1

FUNCTIONS OF THE MATHEMATICAL LIBRARY

Exponential	Hyperbolic Tangent
Natural Logarithm	Hyperbolic Sine
Common Logarithm	Hyperbolic Cosine
Arcsine	Error Function
Arccosine	Gamma Function
Arctangent	Modular Arithmetic
Sine	Absolute Value
Cosine	Truncation
Tangent	Largest Value
Square Root	Smallest Value
Complex Arithmetic	

stored in mass storage devices such as disks, drums, or magnetic tapes. The one disc and six magnetic tape units [40, pages 18, 20] associated with the C8500 are judged to be more than sufficient for the anticipated needs. If the intelligent terminal mode of operation is selected, the two disk units which are presently tied to the GT-44 will provide a more than adequate mass storage facility for the graphics.

The need for a high speed central processing unit is best justified on the grounds of the number and complexity of the calculations to be performed. Some idea of the number of evaluations to be performed for a single functional block in a system block diagram may be obtained from the discussion in Section 2.3, and it should be realized that, for each output data point evaluated, many processing cycles are required.

The final hardware item to be discussed is the interface unit. In the traditional fashion, the interface unit is intended to provide a link between the host computer and its peripherals such as a printer. In the current MIDACS formulations, the interface will additionally be required to provide the link to the graphics terminal. For the situation where the host computer is the C8500 and the graphics terminal is a GT-44, the details of the interface connection are provided in Figure 6-3. One other interface connection which is desirable is indicated in Figure 6-2: the C8500 to a remote programming unit interface. The purpose of this connection is to allow the organization doing the MIDACS software development, assumed to be at a remote location, access to the C8500. This is particularly important when one realizes that the C8500 system is not in wide use and that very few organizations would have the system available for on-site program development.

7. THE DEMONSTRATION PROGRAM

7.1 Illustrating the Concepts

The MIDACS demonstration program was designed to illustrate many of the concepts discussed in the Interim Report and elsewhere in this report. It shows the types of features which can be incorporated in a fully developed interactive modeling system and provides a means for evaluating the relative merits of the various concepts. The demonstration highlights the following features of the interactive modeling system: (1) user-computer interactions are tailored for both experienced and inexperienced users, (2) the light pen is used to direct program flow, (3) the keyboard is used to direct program flow and input data, (4) two levels of communications analysis, system and function, are available, (5) modeling flexibility allows the user some latitude in his decisions, (6) built-in error checking sequences prevent mistakes, and (7) both CRT graphics and teletype style outputs present calculated results of analyses. Each of these concepts along with an example (where available) extracted from a demonstration run is discussed in more detail below.

Three types of user-computer interactions take place in the demonstration program. At many points in the program the computer queries the user for a particular piece of information and waits for the appropriate response, as shown in Figure 7-1. This method of prompting the user for necessary information is suitable for the inexperienced user who is not always sure about what he should do next in exercising the system. At other places in the program the computer merely indicates that it is waiting for input by issuing a prompting character. Here the user must decide what course of action to pursue next, and he uses a pre-defined command to direct the program flow. An experienced user who is familiar with the command structure can use this method of interaction to rapidly formulate and evaluate a problem. In the example of Figure 7-2, the user has specified a signal generator and its characteristics, exercised the signal generator, and generated a graphical output using only three command inputs.

```
MIDACS DEMONSTRATION PROGRAM

PROJECT A-1750      JULY,1976

TO SKIP INTRO, TYPE 'S' AND <CR>
S

.

ARE YOU ON A GT-44 SYSTEM? ANSWER YES OR NO USING THE KEYBOARD.
Y

DO YOU NEED ASSISTANCE IN USING MIDACS?
ANSWER YES OR NO AS BEFORE.
Y
```

Figure 7-1. Example Run Showing Queries and User Responses.

```
*
? SIGGEN,0,2,1000,0,20
*
? PROCES,1
 PROCESSING COMPLETE THROUGH BLOCK  1
*
? TIMPLT
 ENTER START,STOP, AND JUMP INDICES
? 1,40,1
      +

 TIMESTEP=   .5000E-04
              AMPLITUDE: MIN -.2000E+01, MAX  .2000E+01 VOLTS
      +
    0   .1   .2   .3   .4   .5   .6   .7   .8   .9  1.0
    I----I----I----I----I----I----I----I----I----I----I
  1 I                                *
  2 I                                       *
  3 I                                            *
  4 I                                                *
  5 I                                                 *
  6 I                                                *
  7 I                                            *
  8 I                                       *
  9 I                                *
 10 I                        *
 11 I                *
 12 I         *
 13 I    *
 14 I*
 15 *
 16 I*
 17 I    *
 18 I         *
 19 I                *
 20 I                        *
 21 I                                *
 22 I                                       *
 23 I                                            *
 24 I                                                *
 25 I                                                 *
 26 I                                                *
 27 I                                            *
 28 I                                       *
 29 I                                *
 30 I                        *
 31 I                *
 32 I         *
 33 I    *
 34 I*
 35 *
 36 I*
 37 I    *
 38 I         *
 39 I                *
 40 I                        *
    N
*
```

Figure 7-2. User-Computer Interactions and Results.

Finally the computer can output, at the user's option, informative text describing the upcoming parts of the program as shown in Figure 7-3. This convenient method of system documentation eliminates the need for thumbing through the pages of a reference manual to get clarification on certain aspects of the system. The user does not have to get up from the session or have his attention diverted needlessly.

Light pen interactions provide a fast, efficient method of control since the user does not have to divert his attention from the display screen. The light pen operations demonstrated involve primarily the control of program flow by selecting items in a menu list. The menu list provides the names of nine types of communications systems from which the user chooses the system of interest. Depending on the choice made, the computer transfers control to the appropriate sequence for displaying the system diagram and performing the analysis. The light pen is also used to respond to queries by selecting one of the appropriate responses, 'YES' or 'NO', displayed on the screen.

The keyboard is used for both data entry and program control. As described above, the computer either queries the user for inputs or it solicits command inputs using a prompting character. The command language format allows the user some freedom of direction in exercising the system; that is, the command inputs do not have to be in any particular order. In the example run of Figure 7-4 the user has given the 'CASK' command to initiate the system level analysis of a <u>C</u>oherent <u>ASK</u> system. Upon receiving the '/' prompting character, he uses the 'OLD' command to retrieve from permanent storage a set of system parameters. With the 'LIST' and 'CHANGE' commands he edits the set of parameters and then makes a permanent copy of the new data with the 'SAVE' command.

The two levels of analysis available in the demonstration are referred to as the function level and the system level, and sample runs of each are illustrated in Figures 7-2 and 7-4, respectively. At the function level the user can configure the functional elements of a system using the command language format. He may also alter the parameters associated with any element of the configured system, as shown in Figure 7-5. This facility

```
THERE ARE CURRENTLY TWO TYPES OF ANALYSIS IN THE
MIDACS PROGRAM. IN THE SYSTEM LEVEL THE SYSTEM DIAGRAM
IS ALREADY SPECIFIED AND THE PARAMETERS OF INTEREST ARE STUDIED.
AT THE SUBFUNCTION LEVEL THE USER CONSTRUCTS A SYSTEM OR
PARTIAL SYSTEM FROM A SET OF SUBFUNCTION BLOCKS.

YOU SHOULD NOW USE THE LIGHT PEN TO SELECT WHICH TYPE
OF ANALYSIS YOU WANT. THERE WILL BE A TIME DELAY BEFORE
THE LIST APPEARS ON THE SCREEN.  A CTRL C WILL ALWAYS TERMINATE
THE PROGRAM EXECUTION

PAUSE -- TRY RE OR RSU TO CONT
```

Figure 7-3. Sample of Text Information Available to User as Result of "Yes" Response to Last Question in Figure 7-1.

```
*
CASK


            SYSTEM LEVEL ANALYSIS FOR BINARY  ASK
ASSUMPTIONS: 1.      COHERENT DETECTION
             2. ADDITIVE WHITE GAUSSIAN NOISE
/
OLD

/
LIST


  SYSTEM IS                          ASK
1. MARK PROBABILITY                  0.50
2. MESSAGE RATE               0.1000E+05 /SEC
3. XMTR POWER                 0.1000E-02 WATTS
4. XMTR ANT. GAIN                   1.00 DB
5. FREQUENCY                  0.1000E+09 HZ
6. RCVR-XMTR SEPARATION 0.1000E+08 METERS
7. RCVR ANT. GAIN                   1.00 DB
8. NOISE FIGURE                     1.00 DB
9. DECISION THRESHOLD               0.50
/
CHANGE,4,10

/
CHANGE,6,6.E4,\,\

/
CHANGE,7,10

/
CHANGE,8,8

/
CHANGE,3,2

/
LIST


  SYSTEM IS                          ASK
1. MARK PROBABILITY                  0.50
2. MESSAGE RATE               0.1000E+05 /SEC
3. XMTR POWER                 0.2002E+01 WATTS
4. XMTR ANT. GAIN                  10.01 DB
5. FREQUENCY                  0.1000E+09 HZ
6. RCVR-XMTR SEPARATION 0.5994E+05 METERS
7. RCVR ANT. GAIN                  10.01 DB
8. NOISE FIGURE                     8.01 DB
9. DECISION THRESHOLD               0.50
/
SAVE
```

This page, although not completely legible, is included for information purposes only.

Figure 7-4. Sample Run Illustrating Some System Level Commands and Their Results.

```
*
? MODIFY,1
 BLOCK  1 IS A SIGNAL GENERATOR
 1. TYPE= 0 ( 0=SINE, 1=SQUARE WAVE )
 2. AMPLITUDE=  .2000E+01 VOLTS
 3. FREQUENCY=  .1000E+04 HZ
 4. DC OFFSET= 0.        VOLTS
 5. SAMPLES PER PERIOD=  20
 DO YOU WANT TO MAKE CHANGES?
? Y
? 1,1
 1. TYPE= 1 ( 0=SINE, 1=SQUARE WAVE )
 2. AMPLITUDE=  .2000E+01 VOLTS
 3. FREQUENCY=  .1000E+04 HZ
 4. DC OFFSET= 0.        VOLTS
 5. SAMPLES PER PERIOD=  20
 DO YOU WANT TO MAKE CHANGES?
? NO
*
? LPF,1,1500
*
? MODIFY,2
 BLOCK  2 IS A LOW PASS FILTER
 1. TYPE= 1 ( 1=RC LPF, 2=2ND ORDER BUTTERWORTH )
 2. CORNER FREQUENCY=  .1500E+04 HZ
 DO YOU WANT TO MAKE CHANGES?
? NO
*
```

Figure 7-5. Editing Function Parameters.

```
*
? SAVE
 INVALID COMMAND IN SUBFUNCTION ANALYSIS
*
? SMPHLD
 SORRY, THAT SUBROUTINE HAS NOT BEEN WRITTEN YET
*
? BLURP
 UNDEFINED STATEMENT
*
? END OF JOB
      .407 CP SECONDS EXECUTION TIME
```

Figure 7-6. Sample Error Messages at Function Level.

as well as the corresponding facility for the system level analysis (Figure 7-4) contribute to the flexibility of the modeling structure. The system level analysis is based on generic types of systems and uses a set of parameters describing the characteristics of the system as a whole rather than each element comprising the system. This is readily seen by comparing the parameter lists in Figures 7-4 and 7-5. By providing these two approaches to modeling communications systems, the MIDACS program lends itself well to more types of problem solving activities.

Error-checking sequences are another feature of the demonstration program. In Figure 7-6 we see that the command decoder checks for mistakes at the keyboard and outputs an appropriate diagnostic message. Program execution is not affected; and as soon as a valid input is received, the program continues. In the example the 'END OF JOB' command terminated the run and returned control to the host operating system. Similar diagnostics exist for parameter entry and system level commands as well, but it must be remembered that this facility is not completely developed in the demonstration package. Some entry errors may cause the operating system to terminate the run. However, in a fully developed modeling system the error checking facilities would be designed to account for all foreseeable entry mistakes or illogical sequences that may be input by the user in attempting to use the system.

Graphical outputs, where feasible, provide the user a means of quickly evaluating the performance measures and other characteristics that he is studying. The analysis portions of the demonstration include routines for formatting the output data for CRT graphical display or teletype style hard copy. Figure 7-1 includes a teletype plot of the signal generator output. In the demonstration all time domain waveform plots are done on the teletype only. The bit-error-rate curves are generated only on the CRT display and utilize permanently stored displays to generate the coordinate axes and text labeling. Having both types of output allows the user to get results rapidly from the CRT display and get permanent records from the teletype terminal for reports or later reference.

7.2 Software Approach for the Demonstration

Although the demonstration was destined to be run on a Digital Equipment Corp. GT-44 system, the initial stages of program development were conducted using the Georgia Tech campus computer, the Control Data Corp. CYBER 74. The programs were later transferred to a GT-44 system for further development and for adaptation to the restricted working memory space. The multi-region overlay facility of the RT-11 operating system was an invaluable aid to implementing the demonstration package in the GT-44 environment.

An overlay system consists of a root segment, memory-resident overlay regions, and the overlay segments stored on the mass storage device (disk, tape, etc.). The root segment contains the main program, common blocks, and transfer addresses. It is always in memory during a run and is never overlaid.

An overlay region is an area of memory reserved at run time to be shared by two or more overlay segments. A separate area of memory is reserved for each region to which segments have been assigned. At run time the overlay handler brings the segments into memory as they are needed. In order to use the overlay facility, certain rules were observed when structuring the program. These rules are given in Chapter 6 of the RT-11 System Reference Manual [44].

A three region overlay structure was used to implement the software in the 16K environment. The overlay structure generally follows the hierarchical nature of the program flow, as illustrated in Figures 7-7 through 7-12. The overall control program resides permanently in the root segment and can access any subroutine in any overlay segment. It primarily directs program flow to one of the four secondary controlling programs which share the first overlay region, as shown in Figure 7-7. The general text information is allocated to various subroutines in both the first and second overlay regions.

The subfunction processing controller directs the simulation of the configured subfunction model by swapping the subfunction simulation routines into the second overlay region in the proper sequence, as shown in Figure 7-8.

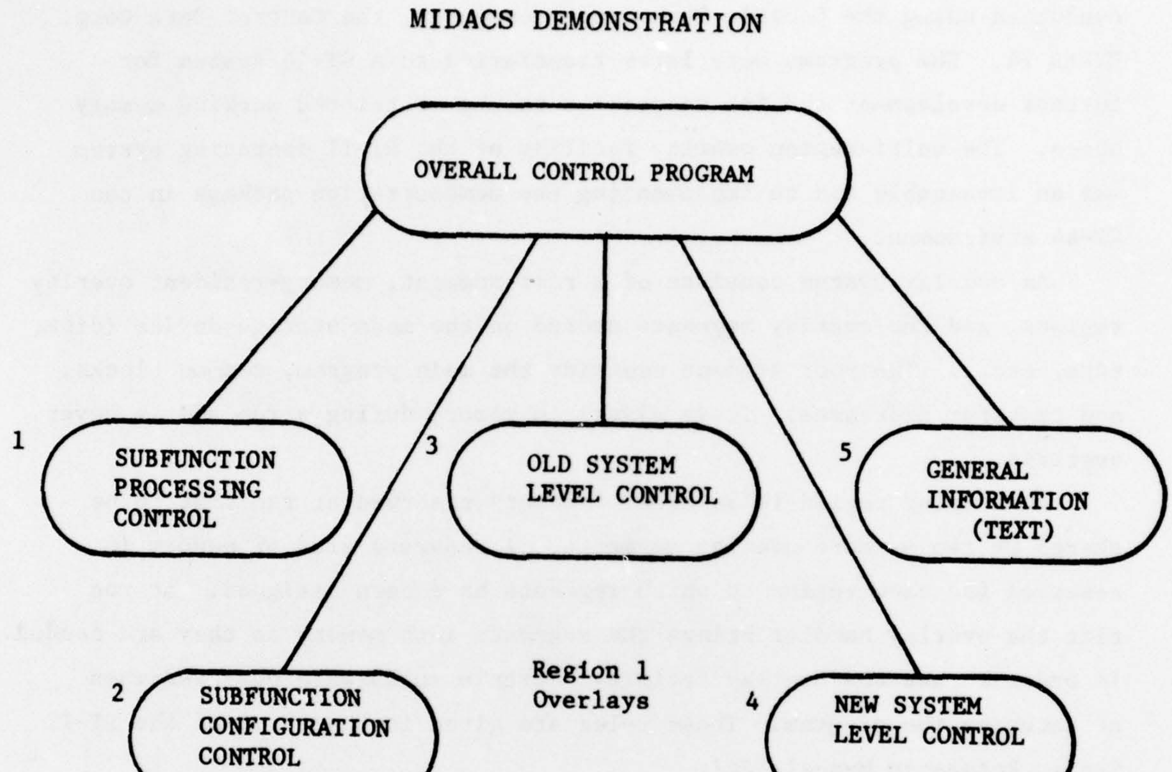


Figure 7-7. Root Segment and Region 1 Overlays 1 through 4.

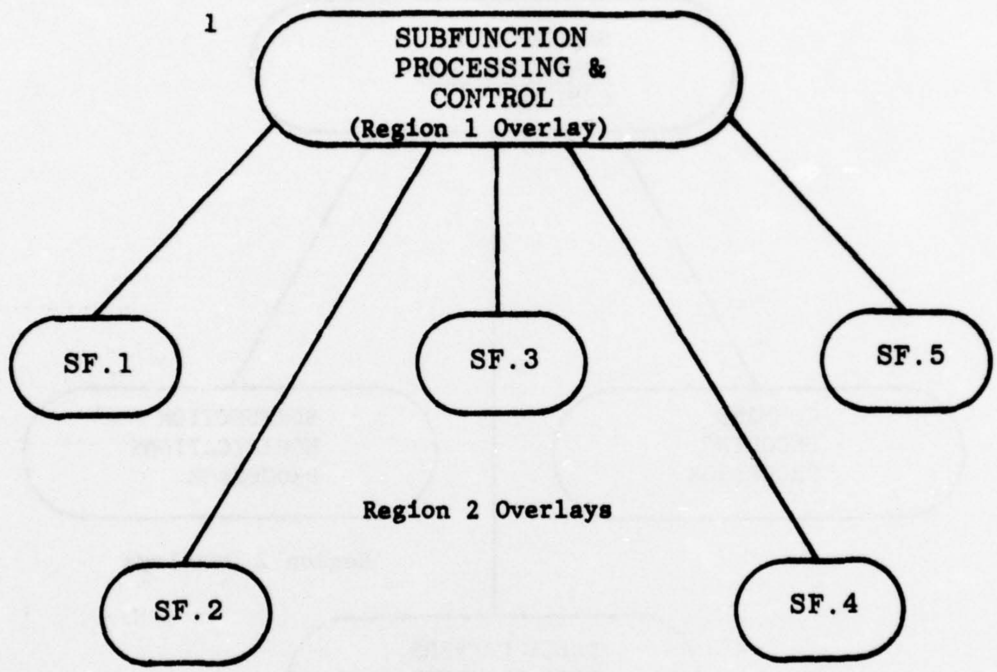


Figure 7-8. Function Elements in the Overlay Structure.

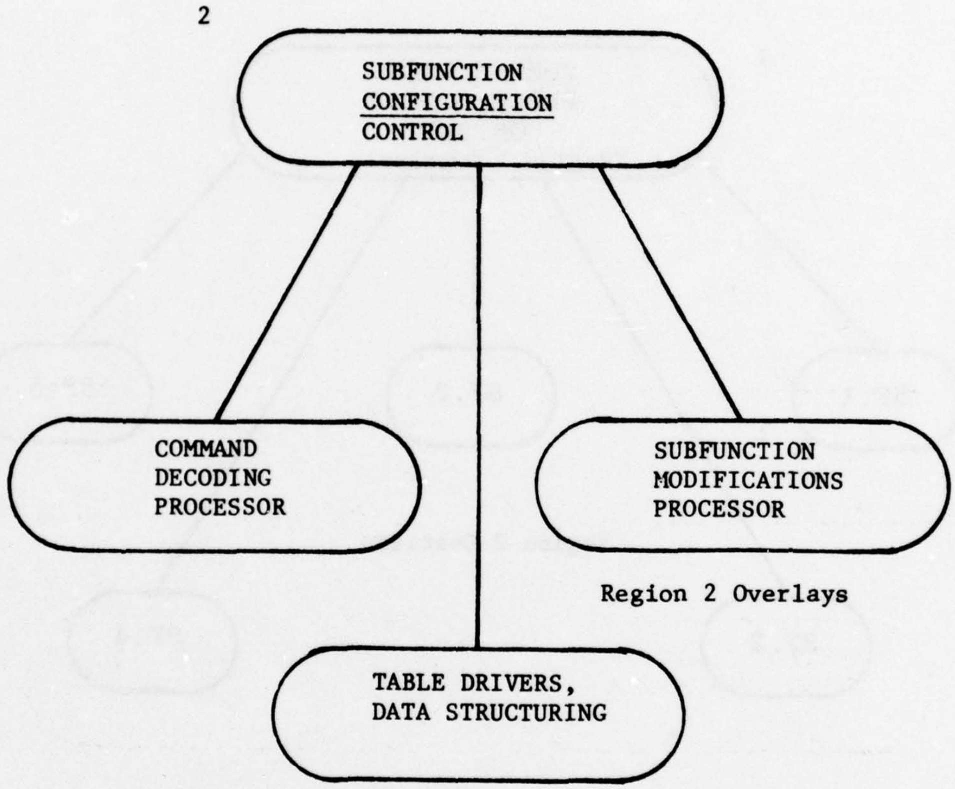


Figure 7-9. Function Configuration Routines.

3

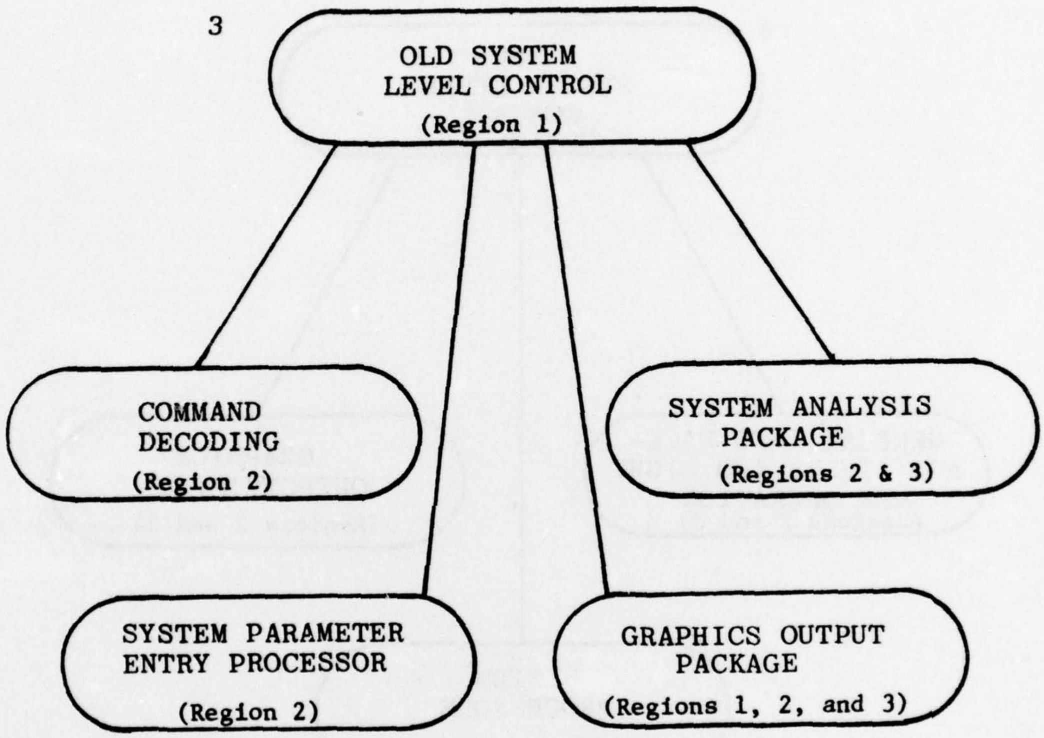


Figure 7-10. Old Version System Level Package.

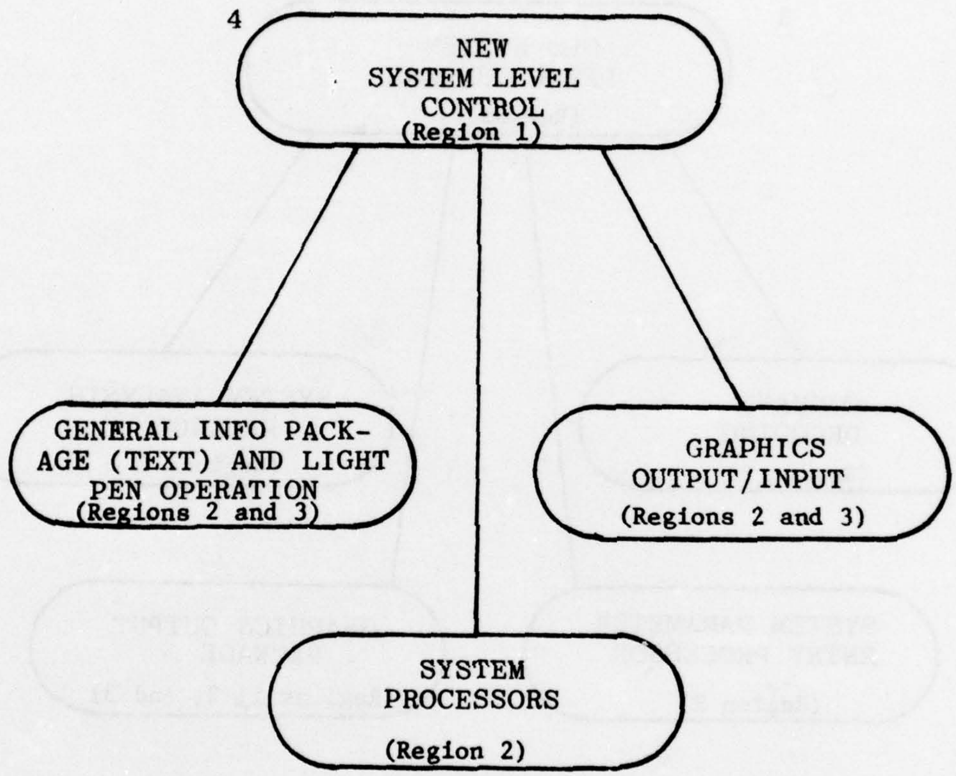


Figure 7-11. New Version System Level Package As Used in MIDSYS (see Appendix C).

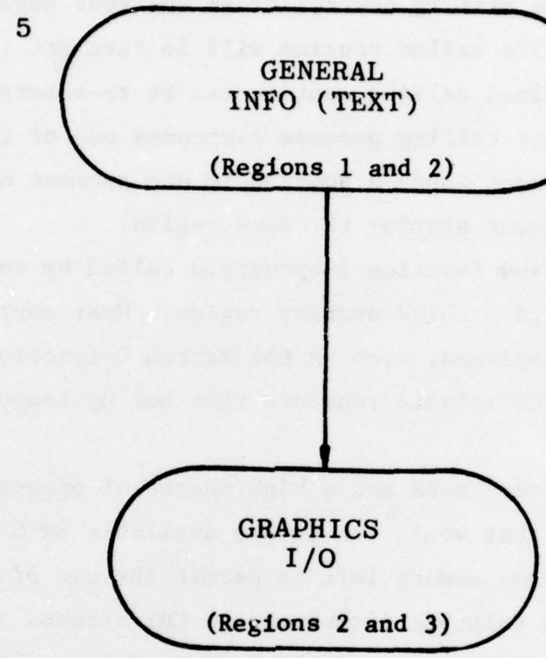


Figure 7-12. Structure for Text and Graphics I/O for Inexperienced Users.

The actual operation of this controller, PRCTRL, is flow charted in Appendix A on page A-81. Similarly, the other secondary controllers operate by calling routines into the secondary overlay region, as shown in Figures 7-9 through 7-12.

There are a few cases where a region 1 routine calls a subroutine in another region 1 segment. In these cases the routine uses a pair of control variables to direct the primary controller in the root segment to call the appropriate routine. The called routine will in turn set the control variables so that the original calling routine can be re-entered at the correct location. This indirect calling process overcomes one of the restrictions of the overlay environment where a routine in one segment may not call a routine in another segment sharing the same region.

Several routines and function subprograms called by routines in region 2 are assigned to a third overlay region. Most routines in region 3 are mathematical algorithms, such as the Marcum Q-function evaluation. There are a few graphics related routines that set up temporary display files in region 3.

The three region structure and a high degree of program segmentation resulted in a program that would run in the available machine, although there was not enough free memory left to permit the use of the GT scrolling feature for displaying teletype output on the CRT screen. In order to demonstrate this feature a subset of the demonstration subroutines was pulled out and a new program created that performs the system level analysis only. Additional details on this program can be found in Appendix C.

7.3 Evaluation of the Demonstration

The demonstration program provided a means for evaluating the applicability of a variety of concepts to MIDACS. Those concepts that worked out well include: the high degree of user-computer interaction, the use of the light pen for program control, the function level and system level approaches to analysis, modeling flexibility, built-in error checking, and CRT graphics I/O. Each of these features is highly desirable for MIDACS and would, of course, be implemented on a large scale in any future program.

Two concepts that did not work out well and should be reconsidered as to their importance were the use of the keyboard and the use of the teletype style graphical outputs. The idea of using a command language is very applicable to an interactive system, especially for the experienced user. However, it would be better to implement the command forms using the CRT/light pen combination. This eliminates a lot of activity at the keyboard and speeds up the operation of the system. Also, the user would no longer be required to remember the formats for the various commands.

The teletype plots can take up a significant amount of time, time in which the user can lose a train of thought or otherwise become distracted. Probably it would be a good idea to have this capability in MIDACS, but only as an option for the user to take advantage of when necessary. In general, the demonstration program showed that operations involving the teletype terminal should be minimized by structuring the program to utilize the CRT and light pen features to the fullest practical extent. The teletype might still be useful as a function keyboard where a single keystroke would result in some action, as does the function keyboard for the OLPARS facility at RADC.

8. RECOMMENDATIONS

8.1 Applications Survey

The initial phase of the MIDACS program has been directed toward identifying and developing the basic concepts involved in an interactive simulation effort. It is anticipated that any future efforts on the MIDACS program will be oriented toward the application of the basic concepts to produce a working simulation program. The value of the implemented program will be strongly tied to the degree to which that program reflects the interests and needs of its prospective users. It is therefore recommended that a survey of prospective users be conducted for the purpose of determining what they would like to see in the simulation program.

In the survey, the prospective users should identify what system types are of the greatest interest in their work. This identification should be as detailed as possible, and the following features should certainly be identified:

1. Name normally used to describe the system,
2. Type of data encoding employed (if any),
3. Type of channel encoding (modulation) employed,
4. Type of transmitter antenna used (if any),
5. Type of channel,
6. Type of receiver antenna used (if any),
7. Type of receiver detection, and
8. Type of decision circuitry.

The information acquired on items two through eight would provide a good indication of the types of systems and functions that should be incorporated into the simulation program in its early phases. Item one may at first seem trivial, but it would provide the basis for a program terminology which would be in close agreement with the already existing system descriptions used by military personnel.

8.2 Modeling Approach

The system level models for MIDACS should be analytical expressions describing a single performance measure in terms of the important system parameters. Closed form representations are highly recommended, but it is recognized that there are many situations in which the only representations that currently exist are the result of the application of infinite series techniques. In such situations every effort should be made to use a truncated series which is consistent with the desired system level attributes of fast evaluation and high accuracy.

The function level models are intended to be used as actual simulation elements, and therefore, the individual models must produce one or more simulated outputs in response to the simulated inputs. Commonly, both inputs and outputs are time-dependent functions; thus, a time-domain representation of the functional blocks is recommended. It is also recognized that there are some circumstances where frequency-domain information is very useful, and a fast Fourier transform (FFT) algorithm in the simulation program is also recommended.

As discussed in Section 2.2, the difference equation and differential equation representations offer the combined features of ease of application and breadth of coverage in modeling communication functions. These two techniques are recommended as the primary representational forms, but this recommendation should not be construed to mean that all functions must be represented in this manner. Any representation which produces the appropriate time-domain output for the specified inputs is usable. This flexibility in representational form should be considered to be an attractive aspect of the chosen simulation structure.

8.3 Simulation Approach

In most instances the system level models may be exercised by simply providing numerical values for the involved parameters. Thus, no special approach is required to carry out a system level exercise or simulation.

The exercise of the function level models results in a true simulation, and therefore, it is necessary to discuss how the simulation will be performed. As pointed out in Section 2.3, certain probabilistic quantities such as probability of bit error and the probability density function of the decision variable are of primary importance. The recommended techniques for evaluating such quantities are the modified Monte Carlo approach and the standard Monte Carlo approach. The modified Monte Carlo scheme is viewed as being the most suitable for an interactive simulation effort.

In recognition of the fact that there are performance measures other than the two mentioned above, it is suggested that additional exercise methods be included in the simulation program. For example, intersymbol interference effects, signal bandwidth, and timing constraints may be best studied in an exercise which does not introduce simulated noise. This scheme has been described in Section 2.3 as a deterministic Monte Carlo approach.

A simulation package capable of performing a standard Monte Carlo, a modified Monte Carlo, and a deterministic Monte Carlo analysis would offer the prospective user a wide variety of evaluation capabilities.

8.4 Programming Structure

It is recommended that the MIDACS software consist of a set of application programs installed in a host computer and in a satellite graphics computer. Each set of programs should be designed to operate under the resident operating systems of the respective computers, specifically, the Disc Oriented Software System (DOSS) for the Collins C8500 and the RT-11 Operating System for the Digital Equipment GT-44. The FORTRAN programming and run time facilities of the two systems are judged to be capable of supporting the development and operation of MIDACS.

The developed software will utilize the resident FORTRAN libraries of each computer system and the RT-11 FORTRAN graphics library. The software should include graphics display routines, analytical and computational programs, a configuration management package including its associated graphics operations, and the controller routines to direct the

program flow in response to user actions during the operation of MIDACS.

Overlay techniques in the GT-44 and program segmentation in the C8500 are recommended so that the available memory is used effectively. Interactions with the user, including user assistance and error messages should be distributed throughout the program structure, and activities of the various program components should be coordinated by a system of control variables maintained in each computer and communicated over the satellite-host link.

8.5 Software Development

The various program components for MIDACS should be designed using the FORTRAN language and possibly the assembly language facilities of the Collins C8500 and the Digital Equipment GT-44. It is likely that much of the host computer software will be developed on a system other than the C8500 due to the latter's unavailability during the initial stages of work. However, by keeping in mind the nature of FORTRAN in the C8500, the program designer can insure that the software will transfer with relatively little modification. The top-down and structured programming approaches should be used to effect a versatile hierarchical framework for MIDACS upon which continuing system expansion can be based. The following schedule is recommended for implementing the minimum system:

1. Establish working hardware configuration,
2. Define communications protocol between computers,
3. Install main controller in host,
4. Install configuration management package in satellite. Test and debug operation of host-satellite system at this stage,
5. Install, test, and debug simulation controller and small set of function simulators in host,
6. Install analytical and computational routines in host,

7. Install graphics display routines in satellite that are associated with analytical routines in host. Test and debug the operation of these sets of programs with the related programs in the host,

8. Use minimum system as a basis for the addition of new function simulators, analytical routines, and associated graphics routines to round out the versatility and applicability of MIDACS to communications problems.

One must remember that adding software almost always requires modifications to the existing software. Proper design and planning insures that the degree of modification is minimized, so that no time is wasted in going back to redesign the program each time new features are incorporated.

8.6 Hardware

A detailed discussion of hardware appears in Section 6. The purpose of the current discussion is to provide an itemized list of recommended hardware, and to that end, the following list is offered:

1. A high speed central processing unit (CPU) with at least 50K of memory and one or more mass storage devices,

2. A lineprinter for producing hard copy of alphanumeric information,

3. A plotter for producing hard copy of graphical information,

4. A graphics terminal with light-pen capabilities and at least 24K of memory and one or more mass storage devices,

5. An interface device capable of coupling the CPU to the graphics terminal and a remote programming terminal, and

6. One or more modem units for communicating with the remote programming terminal.

The Collins C8500 computer with its 64K of memory and associated mass storage devices, which RADC owns, is deemed to be a suitable CPU; however, the DEC GT-44 owned by RADC is inadequate in its present form for extensive graphics development. It is recommended that an additional 16K of memory be acquired to supplement the existing memory.

8.7 Further Development

It is recommended that the concepts discussed in this report be employed in the development of a large scale simulation program. Such a program should be developed in steps, and it is suggested that each step result in an operational software tool. A three step program of the following form is suggested:

Step 1: A preliminary software development effort in which the basic program structure is implemented and a sufficient number of models are developed to allow the evaluation of at least three communication systems,

Step 2: An intermediate software development phase in which the basic program generated in Step 1 is expanded primarily through the development of a large number of system and/or function models, and

Step 3: A review and refinement task in which all of the previously developed software is evaluated on the basis of its utility and efficiency. Also the relationship between the hardware and software will be examined to see if the incorporation of advances in hardware during the period of the program would allow improved program operation.

Based upon the discussion of Section 4, it is suggested that in Step 1 communication systems using line-of-sight, troposcatter, and wire-line channels be modeled. The systems built around these channel models should be capable of a variety of modulation forms and coding types. In particular, provision should be made for modeling FSK, PSK, and DPSK modulation formats and convolutional and block encoding schemes. By employing all of the possible combinations of the features mentioned, one can generate 18 distinct communication systems.

As pointed out elsewhere in this report there are a variety of analysis capabilities that could be incorporated into the program. The following is a list of those capabilities which are judged to be the most important and should be included as a minimum in the simulation program:

1. Time domain waveform generation,

2. Signal-to-noise ratio evaluation,

3. Probability of bit error determination,

4. Signal spectrum characterization, and

5. Probability density function evaluation for the decision variable.

Interactive graphics is central to the simulation concept that has been studied in the current effort and is recommended for any future program development. It is suggested that in any further development the graphics program be capable of:

1. Displaying a complete set of step-by-step instructions on program use at the request of the user,

2. Displaying, structuring, and restructuring the system-level and function-level block diagrams,

3. Accepting light pen inputs of both a qualitative and quantitative nature,

4. Outputting plots of evaluated performance measures where the measure is shown to be a function of a variety of independent variables, and

5. Allowing families of curves to be generated on a single grid structure.

BIBLIOGRAPHY

[1] G. A. Korn, T. M. Korn, Mathematical Handbook for Scientists and Engineers, McGraw-Hill Book Company, New York, 1968.

[2] H. Taub, D. L. Schilling, Principles of Communication Systems, McGraw-Hill Book Company, New York, 1971.

[3] J. B. Thomas, An Introduction to Statistical Communication Theory, John Wiley & Sons, Inc., New York, 1969.

[4] P. Beckmann, Probability in Communication Engineering, Harcovrt, Brace & World, Inc., Atlanta, 1967.

[5] B. P. Lathi, Communication Systems, John Wiley & Sons, Inc., New York, 1968.

[6] S. Stein, J. J. Jones, Modern Communication Principles, McGraw-Hill Book Company, New York, 1967.

[7] H. L. Van Trees, Detection, Estimation, and Modulation Theory, John Wiley and Sons, Inc., New York, 1968.

[8] W. H. Hayt, Jr., J. E. Kemmerly, Engineering Circuit Analysis, McGraw-Hill Book Company, New York, 1962.

[9] E. Kreyszig, Advanced Engineering Mathematics, John Wiley and Sons, Inc., New York, 1967.

[10] E. A. Coddington, An Introduction to Ordinary Differential Equations, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1961.

[11] G. P. Weeg, G. B. Reed, Introduction to Numerical Analysis, Blaisdell Publishing Company, Waltham, Mass., 1966.

[12] P. M. Derusso, R. J. Roy, C. M. Close, State Variables for Engineers, John Wiley & Sons, Inc., New York, 1967.

[13] R. J. Schwartz, B. Friedland, Linear Systems, McGraw-Hill Book Company, New York, 1965.

[14] J. S. Rosko, Digital Simulation of Physical Systems, Addison-Wesley Publishing Company, Reading, Mass., 1972.

[15] R. A. Manske, "Computer Simulation of Narrowband Systems", IEEE Trans. on Computers, Vol. C-17, No. 4, April 1968, pp. 301-308.

[16] V. Castellani, M. Pent, and L. LoPresti, "Multilevel DCPSK Over Real Channels," International Communications Conference, 1974, pp. 39A-1 - 39A-5.

[17] V. Castellani and M. Sant'Agostino, "Intersymbol and Interchannel Interference Effects in PCM Signal Transmission on Radio Links: Binary Coherent Phase Modulation," Alta Frequenza, Vol. 40, May 1971, pp. 109-117.

[18] C. Calavito and M. Sant'Agostino, "Multiple Co-Channel and Interchannel Interference Effects in Binary and Quanternary PSK Radio Systems," International Communications Conference, 1972, pp. 20-6 - 20-11.

[19] D. L. Hedderly and L. Lundquist, "Computer Simulation of a Digital Satellite Communications Link," International Communications Conference, 1972, pp. 2-15 - 2-21.

[20] R. E. Zeimer and S. H. R. Raghavan, "Dispersive Channel Degradation, in Equalized QPSK Data Transmission Systems by Computer Simulation," National Electronics Conference, 1973, pp. 132-137.

[21] J. F. Oberst and D. L. Schilling, "Performance of Self-Synchronized Phase-Shift Keyed Systems," IEEE Trans. on Communication Technology, Vol. COM-17, No. 6, December 1969, pp. 664-669.

[22] S. A. Rhodes and L. C. Palmer, "Computer Simulation of a Digital Satellite Communications System Utilizing TDMA and Coherent Quadriphase Signalling," International Communications Conference, 1972, pp. 34-19 - 34-24.

[23] C. R. Wolfson, "Effects of Phase Jitter on the Probability of Bit Error in Phase-Shift Keyed Systems Employing Error Control Coding," International Communications Conference, 1973, pp. 2-23 - 2-30.

[24] M. C. Austin and G. Hrycenko, "Simulated Performance of Satellite Communications Systems," International Communications Conference, 1975, pp. 20-12 - 20-16.

[25] C. T. Dawson and W. H. Tranter, "SYSTID - A flexible Tool for the Analysis of Communication Systems," National Telecommunications Conference, 1972.

[26] Ernest Freeman and Michael Fashano, "Computer Program Documentation, SYSTID, System Time-Domain Simulation Program," Contract NAS9-10831, System Associates, Long Beach, California, February 1971.

[27] W. H. Tranter, M. J. Fashano, and R. E. Zeimer, "Performance Evaluation Using SYSTID Time Domain Simulation," International Communication Conference, 1975, pp. 20-1 - 20-6.

[28] W. H. Tranter and C. T. Dawson, "Noise Studies of Communications Systems Using the SYSTID Computer Aided Analysis Program", National Electronics Conference, 1973, pp. 114-119.

[29] John M. Wozencrant and Irwin Mark Jacobs, Principles of Communication Engineering, John Wiley & Sons, Inc., New York, 1965, pps. 97-103.

[30] J. D. Foley, "A Tutorial on Satellite Graphics Systems," Computer, August 1976, pp. 14-20.

[31] H. Jasik, Antenna Engineering Handbook, McGraw-Hill Book Company, New York, 1961.

[32] L. A. Berry, G. A. Hufford, "Terrain Effects on Propagation", Work Shop on Radio Systems in Forested and/or Vegetated Environments, Institute of Telecommunication Sciences, Tech. Report No. ACC-ACO-1-74, February 1974.

[33] D. B. Sailors, Calculation of Signal-to-Noise Ratio for HF Propagation Prediction, U. S. Navy Electronics Laboratory Report 1383, 21 June 1966.

[34] Methods of Comparison and Evaluation of Communication Systems, Vol. I, Armour Research Foundation of Ill. Inst. of Tech., RADC-TR-60-97A, 31 March 1960.AD#239198.

[35] D. B. Sailors, A High-Frequency Propagation Prediction Program for the CDC 1604 Computer, U. S. Navy Electronics Laboratory Report 1358m 28 February 1966.

[36] H. M. Smith, "Directory of Computer Programs with Applications to EMC", IEEE Transactions of Electromagnetic Compatibility, pp. 140-141, May, 1974.

[37] J. W. Armstrong, et. al., Alternative Design Approaches for CTGC Transmission-Oriented Subsystems, MITRE Technical Report MTR-2505, 13 March 1973.

[38] E. A. Babineau, et. al., A Comparison of Transmission Systems for Air Force Combat Theater Communications, MITRE Technical Report MTR-1969, 28 October 1970.

[39] Tactical Communications Interface Study, Final Report for Contract F19628-74-C-0017, Electronics Systems Division, Air Force Systems Command, Hanscom Field, March 1975.

[40] N. J. Sturdevant, An Introduction to RADC/DICEF's C8500 Computer System, RADC-TR-74-215, August, 1974.AD#787861/4GI.

[41] Collins Radio Company, C-8560 Time Division Multiplex Loop System, 523-0761359-00171A, 15 January 1969, Cedar Rapids, Iowa.

[42] Digital Equipment Corporation, GT40/GT42 User's Guide, EK-GT40-OP-002, Maynard, Mass.

[43] Digital Equipment Corporation, FORTRAN/RT-11 Extensions Manual, DEC-11-LRTEA-B-D, Maynard, Mass., 1975.

[44] Digital Equipment Corporation, RT-11 System Reference Manual, DEC-11-ORUGA-C-D, Maynard, Mass., 1975.

[45] J. R. Walsh, R. D. Wetherington, L. D. Holland, Study of Mathematical Modeling of Communication Systems Transponders and Receivers, Final Report for NASA Contract NAS8-28148, 19 November 1972.

[46] M. Pent, Electronics Letters, 13 December 1968, page 563.

[47] M. Abramowitz, I. A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards, Applied Mathematics Series 55, December 1965, page 299.

APPENDIX A

MIDACS DEMONSTRATION SOFTWARE DOCUMENTATION

APPENDIX A

MIDACS DEMONSTRATION SOFTWARE DOCUMENTATION

1. General Description

The MIDACS demonstration software is a set of FORTRAN IV programs intended to illustrate some of the concepts formulated for MIDACS. The demonstration was not designed as a complete analytical tool although it can provide useful information for some types of problems. There are two approaches to system level analysis and one somewhat limited function level analysis package. The demonstration allows program control to be exercised either from a keyboard or an interactive graphics terminal, thus providing a ready comparison between these two approaches to interactive modeling and simulation.

The demonstration is designed to run in a Digital Equipment Corp. GT-44 computer system with 16K of main memory. If a larger memory space is available, the text output can be displayed on the CRT rather than the printer. A separate program consisting of a subset of the demonstration software will run in the minimum memory space and allow the display of text on the CRT. The separate program is in file MIDSYS.SAV, and the demonstration program is in file MID16K.SAV on the demonstration DECpack disk cartridge.

Table A-1 is an alphabetical listing of all subroutines developed for the demonstration package. Table A-2 is a file directory showing where the subroutines are located in the disk-resident permanent files. The following sections describe briefly and list the main program and the subroutines as they exist on the GT-44. For a description of the overlay structure used, see Section 7.2. The subroutine calling hierarchy for MIDACS demonstration software is depicted in Appendix D.

TABLE A-1

SUBROUTINES USED IN MIDACS DEMONSTRATION PACKAGE

1. AB1	23. FS2	46. R3YNO
2. AB2	24. GOOF	47. SAHLPF
3. AB3	25. GTPLT	48. SBCTRL
4. AMP	26. GTYCTR	49. SIGGEN
5. ASSIST	27. HINT	50. SLST
6. BCDFPT	28. HOLD	51. SMPHLD
7. CMQF	29. INTRO1	52. SMPLPF
8. CNVT	30. LPF	53. SNR
9. COPY	31. MODIFY	54. SNRF
10. DATIT	32. PE	55. STRDTA
11. DPS	33. PE70	56. SYSAST
12. DRATEF	34. PICT1	57. SYSINT
13. DSPECR	35. PLTEND	58. SYSMSS
14. DSPSNR	36. PRAB	59. SYSTEM
15. ELFIND	38. PRFS	60. SY1MS
16. ENRF	39. PROCES	61. SY30
17. ERF	40. PRPSD	62. SY40
18. ETA	41. PRQ	63. SY50
19. FEAT	42. PS1	64. SY60
20. FEAT1	43. PS2	65. SY70
21. FETCH	44. QNTIZER	66. TIMPLT
22. FS1	45. QPS	67. YNT

TABLE A-2

SUBROUTINE DIRECTORY BY FILE NAME (ON DECpack DISK)

FILENAME (not including extensions)	SUBROUTINES
1. ROOT1	MAIN, DATIT
2. SEG1R1	PRCTRL, PROCES
3. SEG2R1	SBCTRL
4. SEG4R1	SYSTEM
5. SEG6R1	ASSIST
6. SEG7R1	INTRO1
7. SEG8R1	SYSAST
8. R2REM	*Unwritten Subroutines*
9. R2FTBD	FETCH, BCDFPT
10. F2GFTI	TIMPLT, GOOF
11. R2SYSB	SY30, SY40, SY50
12. F2SINT	SYSINT
13. R2MOD	MODIFY
14. R2SIG	SIGGEN
15. R2LPF	COPY, LPF, YNT
16. R2QNT1	QNTIZER, SAHLPF, FEAT1
17. R2SMS1	SY1MS
18. R2SB60	SY60, PE, SNRF, ENRF
19. R2SB70	SY70, PE70
20. R2HINT	HINT

(continued)

TABLE A-2 (continued)

SUBROUTINE DIRECTORY BY FILE NAME (ON DECpack DISK)

FILENAME (not including extensions)		SUBROUTINES
21.	R2SMP	SMPHLD, HOLD, SMPLPF, FEAT
22.	R2GTP	GTPLT
23.	R2SLST	SLST
24.	R2PIK1	PICT1
25.	R2ELST	ELFIND, STRDTA
26.	R2AB1	AB1
27.	R2AB2	AB2
28.	R2AB3	AB3
29.	R2PS1	PS1
30.	R2PS2	PS2
31.	R2QPS	QPS
32.	R2DPS	DPS
33.	R2FS1	FS2
34.	R2FS2	FS2
35.	R2S8MS	SYSMSS
36.	R2PRAB	PRAB
37.	R2PRPD	PRPSD
38.	R2PRQ	PRQ
39.	R2PRFS	PRFS
40.	R2CNVT	CNVT

(continued)

TABLE A-2 (continued)

SUBROUTINE DIRECTORY BY FILE NAME (ON DECpack DISK)

FILENAME (not including extensions)	SUBROUTINES
41. R2SNRP	DSPSNR
42. R2ECRP	DSPECR
43. R3F1	ERF, AMP, ETA
44. R3F2	CMQF
45. R3F3	DRATEF, SNR
46. R3YNO	R3YNO
47. R3GTC	GTYCTR, PLTEND

2. Main Program

Calls: INTRO1, ASSIST, SBCTRL, SYSTEM, PRCTRL, GOOF, SYSAST, GTPLT

Commons: blank, DF1012, CDATA, CCIRKT, CWORD, CDATIN, CRESLT, CBER, CBINRY

Description: MAIN is the overall control program which directs the operations of the MIDACS demonstration. It occupies the root segment in the overlay structure and serves as a switching function to call appropriate region 1 subroutines. It allows subroutines in one overlay segment to call subroutines in other segments via the control variables ICOM and ICNTRL and the computed GO TO statement.

```
0001           PROGRAM MAIN
0002           COMMON/DF1012/IREC,NREC
0003           COMMON/HSTORY/A(10,4)
0004           COMMON T,J,IBLK,IQCNTR,IGRAFX,ICOM,ICNTRL,IPNTS
0005           COMMON/CDATA/JCTR,DATA(50)
0006           COMMON/CCIRKT/NBLK,ITYP(10,2)
0007           COMMON/CWORD/WORD(10)
0008           COMMON/CDATIN/DATAIN(100),DPAST1,DPAST2
0009           COMMON/CRESLT/RESULT(100),RPAST1,RPAST2
0010           COMMON/CBER/BER(40)/CBINRY/BINARY(100),IQWORD(5)
0011  D        DEFINE FILE 10 (3,50,U,NREC)
0012  D        DEFINE FILE 12 (5120,40,U,IREC)
0013           DATA IVAR/1HY/
0014           ICNTRL=0
0015           ICOM=1
0016           CALL INTRO1(GTST)
0017           IF(ICNTRL.EQ.5) CALL ASSIST
0019         1 GO TO (10,20,10,10,50,10,70,80,90,100,110,120,130,140) ICOM
0020        10 CALL SBCTRL(ISYSTM)
0021           GO TO 1
0022        20 CALL SYSTEM(ISYSTM)
0023           GO TO 1
0024        50 CALL PRCTRL
0025           GO TO 2
0026        70 ICNTRL=1
0027           ICOM=1
0028           GO TO 1
0029        80 ICNTRL=2
0030           ICOM=2
0031           GO TO 1
0032        90 CALL GOOF
0033           GO TO 1
0034       100 CALL SYSAST
0035           GO TO 2
0036       110 CALL GOOF
0037           GO TO 1
0038       120 CALL GTPLT
0039           GO TO 1
0040       130 CALL GOOF
0041         2 ICNTRL=0
0042           GO TO 1
0043       140 WRITE(6,8400)
0044      8400 FORMAT(/' THE MIDACS DEMONSTRATION PROGRAM WILL NOW',
              +' TERMINATE',/' AND CONTROL WILL RETURN TO THE RT-11',
              +' MONITOR.')
0045           STOP
0046           END
```

3. Subroutines AB1, AB2, AB3, PS1, PS2, QPS, DPS, FS1, FS2

Calls: RT-11 GT Graphics Subroutines

Called by: SYSAST

Commons: None

Each subroutine sets up a display file in a region 2 overlay and restores into the display the block diagram of the system under consideration as outlined in Table A-3. It then suspends program execution to allow the user as much time as he wants to inspect the diagram. A carriage return keystroke is sufficient to continue program execution.

TABLE A-3

DISPLAY FILES

SUBROUTINE	ASSOCIATED SYSTEM
AB1	ASK with coherent detection
AB2	ASK with non-coherent detection
AB3	ASK with correlation receiver
PS1	PSK with coherent detection
PS2	PSK with correlation receiver
QPS	QPSK (4-phase PSK)
DPS	DPSK (differential PSK)
FS1	FSK with coherent detection
FS2	FSK with non-coherent detection

```
*LIST1
FORTRAN IV        V01C-03A    WED 25-AUG-76 13:35:32                  PAGE 001

      SUBROUTINE AB1
      DIMENSION IBUF(400)
      CALL INIT (IBUF,400)
      CALL SCROL(5,900,5)
      CALL RSTR ('ASKB1')
      PAUSE 'TYPE RE TO CONTINUE'
      CALL FREE
      RETURN
      END
```

```
*PROPRT/X

$JOB

$RT11

ADJUST PAPER AND PRESS <RETURN> KEY
FORTRAN IV      V01C-03A   FRI 30-JUL-76 03:23:13                    PAGE 001

        SUBROUTINE AB2
        DIMENSION IBUF(400)
        CALL INIT (IBUF,400)
        CALL SCROL(5,900,5)
        CALL RSTR ('ASKB2')
        PAUSE 'TYPE RE TO CONTINUE'
        CALL FREE
        RETURN
        END
```

```
FORTRAN IV      V01C-03A   FRI 30-JUL-76 03:23:25              PAGE 001

        SUBROUTINE AB3
        DIMENSION IBUF(400)
        CALL INIT (IBUF,400)
        CALL SCROL(5,900,5)
        CALL RSTR ('ASKB3')
        PAUSE 'TYPE RE TO CONTINUE'
        CALL FREE
        RETURN
        END
```

```
FORTRAN IV      V01C-03A    FRI 30-JUL-76 03:23:58                PAGE 001

      SUBROUTINE PS1
      DIMENSION IBUF(400)
      CALL INIT (IBUF,400)
      CALL SCROL(5,900,5)
      CALL RSTR ('PSKB1')
      PAUSE 'TYPE RE TO CONTINUE'
      CALL FREE
      RETURN
      END
```

```
0001      SUBROUTINE PS2
0002      DIMENSION IBUF(400)
0003      CALL INIT(IBUF,400)
0004      CALL SCROL(5,900,5)
0005      CALL RSTR('PSKB2')
0006      PAUSE 'TYPE RE TO CONTINUE'
0007      CALL FREE
0008      RETURN
0009      END
```

```
FORTRAN IV      V01C-03A   FRI 30-JUL-76 03:24:50                 PAGE 001

      SUBROUTINE QPS
      DIMENSION IBUF(400)
      CALL INIT(IBUF,400)
      CALL SCROL(5,900,5)
      CALL RSTR('QPSKB1')
      PAUSE 'TYPE RE TO CONTINUE'
      CALL FREE
      RETURN
      END
```

```
      SUBROUTINE DPS
      DIMENSION IBUF(400)
      CALL INIT (IBUF,400)
      CALL SCROL(5,900,5)
      CALL RSTR ('DPSKB1')
      PAUSE 'TYPE RE TO CONTINUE'
      CALL FREE
      RETURN
      END
```

```
FORTRAN IV      V01C-03A   FRI 30-JUL-76 03:25:04                PAGE 001

      SUBROUTINE FS1
      DIMENSION IBUF(400)
      CALL INIT (IBUF,400)
      CALL SCROL(5,900,5)
      CALL RSTR ('FSKB1')
      PAUSE 'TYPE RE TO CONTINUE'
      CALL FREE
      RETURN
      END
```

```
FORTRAN IV      V01C-03A   FRI 30-JUL-76 03:25:25                    PAGE 001

      SUBROUTINE FS2
      DIMENSION IBUF(400)
      CALL INIT (IBUF,400)
      CALL SCROL(5,900,5)
      CALL RSTR ('FSKB2')
      PAUSE 'TYPE RE TO CONTINUE'
      CALL FREE
      RETURN
      END
```

4. Function AMP

Calls: None

Called by: ENRF, SNRF

Commons: None

AMP calculates the amplitude of the received signal based on parameters in the data array A which is associated with the old version system level analysis. The evaluation is carried out using the formula below. AMP is used in computing the signal or energy-to-noise ratio for specific values of transmitter power, antenna gain, etc.

$$\text{Amplitude} = \frac{c}{4\pi f R_o} \sqrt{P_t G_t G_r} \;,$$

where c = velocity of light

f = operating frequency in Hz

R_o = transmitter receiver separation distance in meters

P_t = transmitter power in watts

G_t = transmitter antenna gain (converted from dB to linear)

G_r = receiver antenna gain (converted from dB to linear).

FORTRAN IV V01C-03A FRI 30-JUL-76 03:32:10 PAGE 001

```
      FUNCTION AMP(A)
      DIMENSION A(10)
      GT=10.**(A(4)/10.)
      GR=10.**(A(7)/10.)
      PT=A(3)
      RO=A(6)
      F=A(5)
      AMP=(2.38565E7)*SQRT(PT*GT*GR)/(F*RO)
      RETURN
      END
```

5. Subroutine ASSIST

Calls: PICT1

Called by: MAIN

Commons: blank

ASSIST provides the inexperienced user a brief description of the types of problems the MIDACS demonstration will handle. It then provides a light pen sensitive choice of analysis levels through subroutine PICT1.

```
*LST4
FORTRAN IV      V01C-03A   FRI 30-JUL-76 02:15:59                  PAGE 001

0001          SUBROUTINE ASSIST
0002          COMMON T,J,IBLK,IQCNTR,IGRAFX,ICOM,ICNTRL
0003          WRITE(6,100)
0004      100 FORMAT(///' THERE ARE CURRENTLY TWO TYPES OF ANALYSIS IN THE',
             +/' MIDACS PROGRAM. IN THE SYSTEM LEVEL THE SYSTEM DIAGRAM ',
             +/' IS ALREADY SPECIFIED AND THE PARAMETERS OF INTEREST ARE ',
             +'STUDIED.',
             +/' AT THE SUBFUNCTION LEVEL THE USER CONSTRUCTS A SYSTEM OR',
             +/' PARTIAL SYSTEM FROM A SET OF SUBFUNCTION BLOCKS.')
0005          WRITE(6,200)
0006      200 FORMAT(///' YOU SHOULD NOW USE THE LIGHT PEN TO SELECT WHICH',
             +' TYPE',
             +/' OF ANALYSIS YOU WANT. THERE WILL BE A TIME DELAY BEFORE',
             +/' THE LIST APPEARS ON THE SCREEN.  A CTRL C WILL ALWAYS ',
             +'TERMINATE',
             +/' THE PROGRAM EXECUTION'/)
0007          CALL PICT1(ITST)
0008          IF(ITST.EQ.1) GO TO 10
0010          ICOM=1
0011          RETURN
0012       10 ICOM=10
0013          RETURN
0014          END
```

6. Function BCDFPT

Calls: None

Called by: FETCH

Commons: None

BCDFPT accepts binary coded characters representing numerical quantities and converts them to a real number which is returned through the function name. It is part of the command decoding structure and is used to convert parameters in common array WORD into real number format. This function was adapted from FATCAT [45] with minor modifications.

```
      FUNCTION BCDFPT( BCD,N )
      INTEGER DIGIT, E, PLUS, DECPT, BCD
      LOGICAL EXPFLG, DIGFLG, DECFLG, EXSIGN
      DIMENSION BCD(1),KSIGN(3),INTEGR(3),RESULT(3),DIGIT(10)
      DATA DIGIT / 1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
      DATA PLUS, MINUS, E, DECPT / 1H+, 1H-, 1HE, 1H. /
      EXPFLG=.FALSE.
      DIGFLG=.FALSE.
      DECFLG=.FALSE.
      EXSIGN=.FALSE.
      DO 11 K=1,3
      KSIGN(K)=1
      INTEGR(K)=0
   11 CONTINUE
      NPLART=0
      K=1
      DO 31 I=1,N
      ICHAR=BCD(I)
      IF ( ICHAR-PLUS ) 13,23,13
   13 IF ( ICHAR-MINUS ) 14,24,14
   14 DO 15 J=1,10
      IF ( ICHAR-DIGIT(J) ) 15,25,15
   15 CONTINUE
      IF ( ICHAR-DECPT ) 16,26,16
   16 IF ( ICHAR-E ) 21,29,21
   23 IF ( DIGFLG ) GO TO 28
      GO TO 31
   24 IF ( DIGFLG ) GO TO 27
      KSIGN(1)=-1
      GO TO 31
   25 INTEGR(K)=10*INTEGR(K)+J-1
      NPLART=NPLART+K-1
      DIGFLG=.TRUE.
      GO TO 31
   26 IF ( DECFLG ) GO TO 21
      IF ( EXPFLG ) GO TO 21
      DECFLG=.TRUE.
      K=2
      GO TO 31
   27 KSIGN(3)=-1
   28 IF ( EXSIGN ) GO TO 21
      EXSIGN=.TRUE.
      GO TO 30
   29 IF ( EXPFLG ) GO TO 21
      IF ( .NOT. DIGFLG ) GO TO 21
   30 EXPFLG=.TRUE.
      K=3
      NPLASV=NPLART
   31 CONTINUE
      IF ( EXPFLG ) GO TO 32
      EXPON=1.
      GO TO 35
   32 IF ( NPLART-NPLASV-4 ) 33,33,21
   33 IEXPON=INTEGR(3)*KSIGN(3)
```

```
      IF ( IABS( IEXPON ) -37 ) 34,34,21
34    EXPON=10.**IEXPON
      NPLART=NPLASV
35    RTSHFT=10.**NPLART
      RESULT(1)=FLOAT( INTEGR(1)*KSIGN(1) )
      RESULT(2)=FLOAT( INTEGR(2)*KSIGN(1) ) / RTSHFT
      BCDFPT=( RESULT(1)+RESULT(2) )*EXPON
41    RETURN
21    N=-1
      GO TO 41
      END
```

7. Function CMQF

Calls: None

Called by: PE, PE70

Commands: None

CMQF evaluates the complementary Marcum Q-function using a Laguerre polynomial expansion developed by M. Pent [46]. CMQF is used in the analysis of non-coherent ASK systems. In the analysis below $Q^c(x,y)$ is the complementary Marcum Q-function.

$$Q^c(x,y) = \frac{y^2}{2} e^{-x^2/2} \sum_{n=0}^{\infty} P_n\left(\frac{x^2}{2}, \frac{-y^2}{2}\right),$$

where

$$P_n(\alpha,\beta) = \frac{L_n(\alpha)\beta^n}{(n+1)!}$$

$$P_{n+1}(\alpha,\beta) = \frac{(2n+1-\alpha)\beta}{(n+1)(n+2)} \cdot P_n(\alpha,\beta) - \frac{n\beta^2}{(n+1)^2(n+2)} \cdot P_{n-1}(\alpha,\beta),$$

$$P_o(\alpha,\beta) = 1,$$

and

$$P_1(\alpha,\beta) = (1-\alpha)\beta/2.$$

$L_n(\alpha)$ is the Laguerre polynomial. CMQF uses fifty terms in the series and tests for out-of-range results (number too large or too small to be represented by computer).

FORTRAN IV   V01C-03A   FRI 30-JUL-76 03:32:53   PAGE 001

```
      FUNCTION CMQF(X,Y)
      DIMENSION P(50)
      A=(Y**2)/2
      B=-(X**2)/2
      IF(ABS(B).GT.67) GO TO 20
      AL=-B
      BT=-A
      P(1)=1.
      P(2)=(1-AL)*BT/2
      SUM=P(1)+P(2)
      DO 30 I=3,50
      C1=FLOAT(I-2)
      C2=(2*C1+1-AL)*BT/((C1+1)*(C1+2))
      C3=C1*(BT**2)/(((C1+1)**2)*(C1+2))
      P(I)=C2*P(I-1)-C3*P(I-2)
      SUM=SUM+P(I)
   30 CONTINUE
      CMQF=A*EXP(B)*SUM
      RETURN
   20 CMQF=0.
      RETURN
      END
```

8. Subroutine CNVT

Called by: SYSAST

Calls: None

Commons: CYC, CUPB

CNVT converts the probability-of-error values in the arrays YC and UPB into absolute screen coordinates for later display on the CRT. The formula is derived from consideration of the area of the screen to be used and the logarithmic scale:

$$\text{Absolute Y coordinate} = 100 + 100(6 + \log(\text{prob. of error})).$$

For a probability of error range from 0 to 10^{-6} this formula gives a screen coordinate range of 700 to 100 in the Y-direction.

```
      SUBROUTINE CNVT
      COMMON/CYC/YC(40)
      COMMON/CUPB/UPB(40),IBND
      DO 10 I=1,40
      IF(YC(I).LT.1.E-6) GO TO 20
   10 YC(I)=100.+100.*(6.+ALOG10(YC(I)))
   20 IF(IBND.EQ.0) RETURN
      DO 30 I=1,40
      IF(UPB(I).LT.1.E-6) RETURN
   30 UPB(I)=100.+100.*(6.+ALOG10(UPB(I)))
      RETURN
      END
```

9. Subroutine COPY

Calls: None

Called by: LPF

Commons: blank, HSTORY, CDATIN, CRESLT

COPY writes the data in the output array RESULT into the input array DATAIN. It also restores into the variables DPAST1, DPAST2, RPAST1, and RPAST2 the last two input values and output values from the last data block processed through the current subfunction element (the one that has called COPY). This past history information is needed in some simulation models to minimize transient effects as data blocks are processed through the model.

```
*LST18
FORTRAN IV        V01C-03A   FRI 30-JUL-76 02:54:54                PAGE 001

      SUBROUTINE COPY
      COMMON/HSTORY/A(10,4)
      COMMON T,I,IBLK
      COMMON/CDATIN/DATAIN(100),DPAST1,DPAST2
      COMMON/CRESLT/RESULT(100),RPAST1,RPAST2
      DO 10 J=1,100
   10 DATAIN(J)=RESULT(J)
      DPAST1=A(IBLK,4)
      DPAST2=A(IBLK,3)
      RPAST1=A(IBLK,2)
      RPAST2=A(IBLK,1)
      RETURN
      END
```

10. Subroutine DATIT

Calls: None

Called by: SBCTRL, SYSTEM

Commons: CWORD

DATIT initializes the WORD array to Hollerith blanks. This subroutine is used instead of simple assignment statements because of some restrictions on Hollerith data in PDP-11 FORTRAN.

```
FORTRAN IV        V01C-03A   FRI 30-JUL-76 02:05:04                    PAGE 001

          SUBROUTINE DATIT
          COMMON/CWORD/WORD(10)
          DATA WORD/10*6H       /
          RETURN
          END
```

11. Function DRATEF

Calls: None

Called by: SY60

Commons: None

DRATEF calculates the data rate from the mark probability and message rate using the formula

$$\text{Data rate} = r\left[-p\log_2 p - (1-p)\,\log_2(1-p)\right],$$

where

r = message rate in messages/sec

p = probability of either mark or space.

```
FORTRAN IV      V01C-03A    FRI 30-JUL-76 03:33:07                PAGE 001

      FUNCTION DRATEF(A)
      DIMENSION A(10)
      P=A(1)
      R=A(2)
      DRATEF=(-R/ALOG(2.))*(P*ALOG(P)+(1.-P)*ALOG(1.-P))
      RETURN
      END
```

12. Subroutines DSPECR, DSPSNR

Calls: Graphics library

Called by: SYSAST

Commons: CYC, CUPB (DSPSNR only)

DSPECR and DSPSNR are the graphics routines for displaying the computed bit-error-rate data on a semi-log grid. In DSPECR the horizontal axis represents energy contrast ratio while in DSPSNR it represents signal-to-noise ratio. Each routine sets up a display file in a Region 2 overlay and restores a previously saved display consisting of a semi-log grid with labeled axes. Then it draws the bit-error-rate curve on the grid using data in arrays YC and UPB (DSPSNR only for the latter).

```
FORTRAN IV        V01C-03A   FRI 30-JUL-76 03:28:29               PAGE 001

           SUBROUTINE DSPECR
           COMMON/CYC/YC(40)
           DIMENSION IBUF(590)
           CALL INIT(IBUF,590)
           CALL SCROL(5,900,5)
           CALL RSTR('EGRID3')
           CALL APNT (200.,YC(1),0,-6)
           DO 70 I=2,40
           IF(YC(I).LT.100.) GO TO 75
        70 CALL VECT (15.,YC(I)-YC(I-1))
        75 PAUSE 'TYPE RE OR RSU TO CONTINUE'
           CALL SCROL(-1,-1,-1)
           CALL FREE
           RETURN
           END
```

```
      SUBROUTINE DSPSNR
      COMMON/CYC/YC(40)
      COMMON/CUPB/UPB(40),IBND
      DIMENSION IBUF(590)
      CALL INIT(IBUF,590)
      CALL SCROL(5,900,5)
      CALL RSTR('EGRID1')
      CALL APNT(200.,YC(1),0,-6)
      DO 70 I=2,40
      IF(YC(I).LT.100.) GO TO 75
   70 CALL VECT (15.,YC(I)-YC(I-1))
   75 IF(IBND.EQ.0) GO TO 90
      CALL APNT (200.,UPB(1),0,-6)
      DO 80 I=2,40
      IF(UPB(I).LT.100.) GO TO 90
   80 CALL VECT (15.,UPB(I)-UPB(I-1))
   90 PAUSE 'TYPE RE OR RSU TO CONTINUE'
      CALL SCROL(-1,-1,-1)
      CALL FREE
      RETURN
      END
```

13. Subroutine ELFIND

Calls: None

Called by: SBCTRL, SYSTEM

Commons: None

ELFIND compares a Hollerith string of up to six characters (four in PDP-11 FORTRAN) to a permanent list of character strings. When a match is found, the integer L is set to a unique value identifying the match. L is then used in the calling program to direct program flow such that the command represented by the character string is executed. This subroutine was adapted from FATCAT [45].

```
0001          SUBROUTINE ELFIND (NAME,L)
0002          REAL NAME,MATCH(45)
0003          DATA MATCH
             +/6HSIGGEN,6HSMPHLD,6HQNTIZR,6HNCODER,4HXMTR   ,
             + 6H     ,4HRCVR   ,6HDECDEV,6HDCODER,3HLPF    ,
             + 3HBPF    ,6HPROCES,6HMODIFY,6HBERPLT,6HTIMPLT,
             + 5HLSTCO  ,6HENDOFJ,4HDEMO   ,6HDVRPLT,6HRESTRT,
             + 6HREPT   ,6HCONTIN,4HCASK   ,4HCPSK   ,4HCFSK   ,
             + 5HNCFSK  ,4HDPSK   ,5HNCASK  ,4HSYS2   ,6H      ,
             + 6H       ,6H      ,6H      ,6H      ,6H      ,
             + 6H       ,6H      ,3HOLD   ,4HLIST   ,3HNEW    ,
             + 6HCHANGE,6H      ,6H      ,4HEXIT   ,4HSAVE    /
0004          DO 11 I=1,45
0005          IF (NAME-MATCH(I)) 11,21,11
0006     11   CONTINUE
0007          I=46
0008     21   L=I
0009          RETURN
0010          END
```

14. Function ENRF

Calls: AMP, ETA

Called by: SY60

Commons: None

ENRF calculates the energy-to-noise ratio from the set of system level parameters according to the formula

$$ENR = 2E/\eta,$$

where $E = A^2T/2$, A is the AMP function,

T = sampling period = 1/message rate,

η = 2 x noise power spectral display,

= function ETA.

```
FORTRAN IV      V01C-03A   FRI 30-JUL-76 03:30:33                PAGE 001

      FUNCTION ENRF(A,K)
      DIMENSION A(10)
      R=A(2)
      ENRF=AMP(A)**2/(R*ETA(A,K))
      RETURN
      END
```

15. Function ERF

Calls: None

Called by: PE, PE70, PRFS, PRAB, PRPSD, PRQ

Commons: None

ERF calculates the error function using the approximation [47]

$$\mathrm{Erf}(x) = 1 - (a_1 t + a_2 t^2 + a_3 t^3)e^{-x^2},$$

where $t = 1/(1+px)$,

$p = 0.47047$,

$a_1 = 0.3480242$,

$a_2 = -0.0958798$,

$a_3 = 0.7478556$.

Accuracy: $|\text{error}| \leq 2.5 \times 10^{-5}$ for $0 \leq x < \infty$

```
FORTRAN IV      V01C-03A   FRI 30-JUL-76 03:32:09                PAGE 001

      FUNCTION ERF(X)
      XT=ABS(X)
      T=1./(1.+.47047*XT)
      Y=-XT**2.
      IF(ABS(Y).GT.675.) GO TO 10
      ERF=1.-(.3480242*T-.0958798*T**2.+.7478556*T**3.)*EXP(-XT**2.)
   20 IF(X.LT.0) ERF=-ERF
      RETURN
   10 ERF=1.
      GO TO 20
      END
```

16. Function ETA

Calls: None

Called by: ENRF, SNRF

ETA computes the value of η in the formulas for signal or energy to noise ratios according to the formulas

$$\eta = kT_e, \text{ if noise temperature is specified or}$$

$$\eta = 290\ k\left[10^{(F_{dB}/10)} - 1\right], \text{ if noise figure is specified,}$$

where k = Boltzmann's constant $\simeq 1.38 \times 10^{-23}$ joule/°K,

T_e = effective noise temperature in Kelvin degrees,

F_{dB} = noise figure in dB.

```
FORTRAN IV      V01C-03A   FRI 30-JUL-76 03:32:12                    PAGE 001

0001            FUNCTION ETA(A,K)
0002            DIMENSION A(10)
0003            RNF=A(8)
0004            ETA=(1.38E-23)*RNF
0005            IF(K.EQ.1)ETA=(4.0038E-21)*(10.**(RNF/10.)-1)
0007            RETURN
0008            END
```

17. Functions FEAT, FEAT1

Calls: None

Called by: SMPHLO, QNTIZER

Commons: blank

FEAT and FEAT1 are identical routines which evaluate a low pass filter constant for 1, 2, or 3 sections of single pole low pass filtering. This allows subfunctions such as the sample-and-hold device to be simulated with band-limiting characteristics included.

For a single section filter F_c is given directly as block bandwidth in Hz.

For a double or triple section filter, F_c is calculated as below:

$$F_c = 2\pi F_c' / \sqrt{\sqrt{2} - 1} \quad \text{(double section)}$$

$$F_c = 2\pi F_c' \sqrt{\sqrt[3]{2} - 1}$$

where F_c' is the desired cutoff frequency for the overall filtering action.

The reason for having two identical subroutines is that they are in overlay segments occupying the same overlay region.

```
FORTRAN IV        V01C-03A    FRI 30-JUL-76 02:55:16                  PAGE 001

0001        FUNCTION FEAT(I,BLKBW)
0002        COMMON T
0003        IF(I.EQ.1) FEAT=EXP(-6.283185*BLKBW*T)
0005        IF(I.EQ.3) FEAT=EXP(-12.3242*BLKBW*T)
0007        IF(I.EQ.2) FEAT=EXP(-9.76262*BLKBW*T)
0009        RETURN
0010        END
```

```
FORTRAN IV      V01C-03A   FRI 30-JUL-76 02:55:30                PAGE 001

      FUNCTION FEAT1(I,BLKBW)
      COMMON T
      IF(I.EQ.1) FEAT=EXP(-6.283185*BLKBW*T)
      IF(I.EQ.3) FEAT=EXP(-12.3242*BLKBW*T)
      IF(I.EQ.2) FEAT=EXP(-9.76262*BLKBW*T)
      RETURN
      END
```

18. Subroutine FETCH

Calls: BCDFPT

Called by: SBCTRL, SYSTEM

Commons: CWORD, CFETCH

FETCH is the command input routine. It reads in commands as a string of characters, decodes the various elements in the input stream, and store them in array WORD. Blank characters are ignored and different elements a delimited by commas. Hollerith strings are truncated to the first six cha acters (four in the PDP-11) and stored in WORD. Numeric characters representing data are converted to real numbers by BCDFPT prior to being saved in WORD. FETCH is adapted from FATCAT [45] with minor changes to accommod the PDP-11 version of FORTRAN.

```
*LST10
FORTRAN IV        V01C-03A   FRI 30-JUL-76 02:30:58                PAGE 001

              SUBROUTINE FETCH (LL,NBAD)
              COMMON/CWORD/ WORD(10)
              INTEGER APOST,BLANK,COMMA,DECPT,EQUAL,PLUS,RPAREN,TEST
              INTEGER BUFF1,BUFF2,BUFF3,BCDFPT,TITLE,WORD*4,E
              DIMENSION TITLE (12)
              COMMON/CFETCH/BUFF2(6),BUFF1(80)
              DATA APOST/1H'/
              DATA BLANK,COMMA,DECPT,EQUAL,MINUS,NINE,NZ,PLUS
             +  / 1H ,1H,,1H.,1H=,1H-,1H9,1H0,1H+ /
              DATA LPAREN,RPAREN,E / 1H(,1H),1HE/
              L=0
              NCOLS=80
              NBAD=1
            1 CONTINUE
              READ (5,1001)  (BUFF1(I),I=1,80)
           20 M=0
            2 K=0
              N=0
              NCOMMA=0
              DO 3 I=1,6
            3 BUFF2(I)=BLANK
            4 CONTINUE
      D       WRITE(6,2200) K
      D2200 FORMAT(' K=',I3)
              IF(M-NCOLS) 40,100,100
           40 M=M+1
              TEST=BUFF1(M)
              IF(TEST-BLANK) 41,4,41
           41 IF(TEST-COMMA) 42,6,42
           42 IF(TEST-EQUAL) 43,6,43
           43 IF(TEST-LPAREN) 44,6,44
          44  IF(TEST-RPAREN) 45,4,45
           45 N=N+1
              BUFF2(N)=TEST
              IF(K) 5,5,4
            5 NCOMMA=1
              IF(TEST-NZ) 52,51,51
           51 IF(TEST-NINE) 4,4,52
           52 IF(TEST-PLUS) 53,4,53
           53 IF(TEST-MINUS) 54,4,54
           54 IF(TEST-APOST) 55,30,55
           55 IF(TEST-DECPT) 56,57,56
           56 IF(TEST-E)561,560,561
          560 IF(N-1) 561,561,4
          561 K=2
              GO TO 4
           57 K=1
              GO TO 4
            6 IF(K-1) 7,7,9
            7 BUFF3=BCDFPT(BUFF2,N)
      D       WRITE(6,2020)BUFF3
      D2020 FORMAT(' BUFF3=' I10,/)
              IF(N) 106,106,8
            8 WORD(L+1)=BUFF3
```

```
      GO TO 91
    9 ENCODE(6,1002,WORD(L+1))   (BUFF2(I),I=1,6)
   91 L=L+1
      IF(M-NCOLS) 2,10,10
   10 IF(TEST-COMMA) 11,1,11
   11 LL=L
D     WRITE(6,555) (WORD(I),I=1,10)
D 555 FORMAT(5I12,/5I12)
      RETURN
   30 ENCODE(72,1001,TITLE) (BUFF1(I),I=1,72)
      GO TO 1
  100 IF(NCOMMA) 6,1,6
  106 WRITE(6,2000)   (BUFF2(I),I=1,6)
      NBAD=0
      RETURN
  107 STOP
 1001 FORMAT(80A1)
 1002 FORMAT(6A1)
 2000 FORMAT(4H0*** ,6A1,20H CANNOT BE DECODED**//)
      END
```

19. Subroutine GOOF

Calls: None

Called by: MAIN, SBCTRL

Commons: None

GOOF is a short diagnostic which is called when certain features or commands are attempted which are not fully implemented in the demonstration software. It prevents error exits and allows the user to go on with something else.

```
FORTRAN IV        V01C-03A    FRI 30-JUL-76 02:32:48                PAGE 001

          SUBROUTINE GOOF
          COMMON T,J,IBLK,IQCNTR,IGRAFX,ICOM,ICNTRL
          ICOM=7
          WRITE(6,100)
      100 FORMAT(' SORRY, THAT SUBROUTINE HAS NOT BEEN WRITTEN YET')
          RETURN
          END
```

20. Subroutine GTPLT

Calls: GTYCTR, PLTEND, Graphics library

Called by: MAIN

Commons: CBER

GTPLT is the plotting routine for the old system level. It performs the same operations as DSPSNR except that it is operating on a different data array. GTPLT is called indirectly from SYSTEM by having SYSTEM set the control variable ICOM before returning to the main program. This calling sequence is a carry-over from the original two-level overlay structure in which SYSTEM and GTPLT were both assigned to the same region. In the three-level structure GTPLT could be called directly.

```
FORTRAN IV      V01C-03A    FRI 30-JUL-76 03:31:30                  PAGE 001

      SUBROUTINE GTPLT
      COMMON/CBER/BER(40)
      DIMENSION IBUF(590)
      CALL INIT (IBUF,590)
      CALL GTYCTR
      CALL SCROL (5,900,5)
      CALL RSTR ('EGRID1')
      CALL APNT (200.,BER(1),0,-5)
      DO 70 I=2,40
      IF(BER(I).LT.100.) GO TO 75
   70 CALL VECT (15.,BER(I)-BER(I-1))
   75 CALL PLTEND
      CALL FREE
      CALL SCROL(-1,-1,-1)
      RETURN
      END
```

21. Subroutine GTYCTR

Calls: None

Called by: GTPLT

Commons: blank, CBER

GTYCTR converts the data in array BER into the corresponding absolute screen y-coordinates, which are used to generate the bit-error-rate curve. The formula is the same as in subroutine CNVT.

```
FORTRAN IV      V01C-03A    FRI 30-JUL-76 03:33:22                    PAGE 001

0001            SUBROUTINE GTYCTR
0002            COMMON T,JJ,IB,IQ,IG,IC,ICN
0003            COMMON/CBER/BER(40)
0004            DO 60 I=1,40
0005            IF(BER(I).LT.1.E-6) GO TO 65
0007         60 BER(I)=100.+100.*(6.+ALOG10(BER(I)))
0008         65 RETURN
0009            END
```

22. Subroutine HINT

Calls: None

Called by: SYSTEM

Commons: HDATA

HINT is used to set up variable text data which is output during the execution of the old version system level analysis routines. The particular text data that is assigned depends on the system under consideration and the values of control variables passed through the argument list. The text data output is saved in common block HDATA.

```
          SUBROUTINE HINT(I,JTYP,IGTST,K)
          COMMON/HDATA/HH(4),HX,HZ(9)
          DIMENSION XARY(4),ZARY(18),HARY(8)
          DATA XARY/2H A,2H P,2H F,2HDP/,
     +          ZARY/4HDECI,4HSION,4H THR,4HESHO,2HLD,3H    ,1HE,
     +               4HNOIS,4HE BA,4HND W,4HIDTH,2H  ,3H HZ,1HS,
     +               4HENER,2HGY,4HSIGN,2HAL/,
     +          HARY/4HFIGU,2HRE,2HDB,4HTEMP,2H. ,4HDEG.,4H    ,4HNON-/
          IF(IGTST.EQ.1) GO TO 500
          GO TO (410,420,430,440,450,460) I
      410 HX=XARY(1)
      411 DO 412 M=1,7
      412 HZ(M)=ZARY(M)
          HH(4)=HARY(7)
          JTYP=0
          GO TO 470
      420 HX=XARY(2)
          GO TO 411
      430 HX=XARY(3)
          GO TO 411
      440 HX=XARY(3)
      441 DO 442 M=1,7
      442 HZ(M)=ZARY(M+7)
          HH(4)=HARY(8)
          JTYP=1
          GO TO 470
      450 HX=XARY(4)
          GO TO 441
      460 HX=XARY(1)
          GO TO 441
      470 HZ(8)=ZARY(15+2*JTYP)
          HZ(9)=ZARY(16+2*JTYP)
          RETURN
      500 LCT=0
          IF(K.EQ.1) LCT=3
          DO 31 M=1,3
       31 HH(M)=HARY(M+LCT)
          RETURN
          END
```

23. Subroutine HOLD

Calls: None

Called by: SMPHLD

Commons: CRESLT

HOLD performs the holding operation of the sample-and-hold device by generating up to twenty data points having the same value as the current input sample point, which is determined by the variable I in the argument list. UPPER and LOWER are bounds on the output determined in SMPHLD from the user-supplied specification list.

```
FORTRAN IV        V01C-03A    FRI 30-JUL-76 02:55:12                    PAGE 001

      SUBROUTINE HOLD(A,UPPER,LOWER,I)
      DIMENSION A(20)
      COMMON/CRESLT/RESULT(100)
      INTEGER UPPER
      DO 10 J=1,20
      A(J)=0
      IF(LOWER.LE.J.AND.J.LE.UPPER) A(J)=RESULT(I)
   10 CONTINUE
      RETURN
      END
```

24. Subroutine INTRO1

Calls: None

Called by: MAIN

Commons: blank

INTRO1 presents the initial introduction to the MIDACS demonstration program. It also sets certain control variables depending on user responses to several questions. These controls are used by other parts of the software to effect proper program flow.

AD-A037 833 GEORGIA INST OF TECH ATLANTA ELECTRONICS TECHNOLOGY LAB F/G 9/2
INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY.(U)
FEB 77 R W MOSS, R W RICE, D R SENTZ F30602-75-C-0247
UNCLASSIFIED RADC-TR-77-42 NL

3 OF 4
AD
A037833

```
0001          SUBROUTINE INTRO1(GTST)
0002          COMMON T,J,IB,IQ,IGRAFX,ICOM,ICNTRL
0003          DATA HY/1HY/ HS/1HS/
0004          WRITE(6,100)
0005      100 FORMAT(/////'                   MIDACS DEMONSTRATION PROGRAM',
         +//'               PROJECT A-1750      JULY,1976')
0006          WRITE(6,110)
0007      110 FORMAT(//' TO SKIP INTRO, TYPE ''S'' AND <CR>')
0008          READ(5,130)X
0009      130 FORMAT(A1)
0010          IF(X.EQ.S)GO TO 1000
0012          WRITE(6,140)
0013      140 FORMAT(//'     THIS PROGRAM DEMONSTRATES SOME OF THE',
         +' CONCEPTS FOR THE',
         +/' MIDACS ( MODELING AND INTERACTIVE DESIGN FOR THE ANALYSIS',
         +/' OF COMMUNICATIONS SYSTEMS ). YOU MAY RESPOND TO THE',
         +/' COMPUTER''S INQUIRIES BY USING THE KEYBOARD AND/OR THE',
         +/' LIGHT PEN.',
         +/'    ALTHOUGH THE DEMONSTRATION PROGRAM CAN DETECT SOME',
         +/' TYPES OF ENTRY ERRORS, YOU SHOULD BE REASONABLY CAREFUL',
         +/' WHEN ENTERING DATA. FEEL FREE TO USE THE DELETE AND',
         +/' CTRL U KEYS TO CORRECT ERRORS BEFORE TYPING THE ',
         +/' CARRIAGE RETURN <CR>.')
0014     1000 WRITE(6,150)
0015      150 FORMAT(//'ARE YOU ON A GT-44 SYSTEM? ANSWER YES OR NO',
         +' USING THE KEYBOARD.')
0016          READ(5,8200)GTST
0017     8200 FORMAT(A1)
0018          IGRAFX=0
0019          IF(GTST.EQ.HY) IGRAFX=1
0021          WRITE(6,8100)
0022     8100 FORMAT(' DO YOU NEED ASSISTANCE IN USING MIDACS?',
         +/' ANSWER YES OR NO AS BEFORE.')
0023          READ(5,8200)GTST
0024          IF(GTST.EQ.HY) ICNTRL=5
0026          RETURN
0027          END
```

25. Subroutine LPF

Calls: COPY, YNT

Called by: PROCES

Commons: blank, CRESLT, CDATA, CDATIN

LPF is the low pass filter simulator for the simple RC filter and the second-order Butterworth filter. The input array is obtained from COPY, and the output is written to array RESULT and locations RPAST1 and RPAST2.

For a simple RC low pass filter with inputs $x(t_n)$ and outputs $y(t_n)$,

$$y(t_{n+1}) = e^{-\alpha T}\, y(t_n) + (1 - e^{-\alpha T})(t_n),$$

where $\alpha = 2\pi F_{cutoff}$

T = sampling period.

For the 2nd order Butterworth case,

$$\text{with } c_1 = e^{-\alpha T/\sqrt{2}}$$

$$c_2 = \cos(\alpha T/\sqrt{2})$$

$$c_3 = \sin(\alpha T/\sqrt{2})$$

we have

$$y(t_{n+1}) = 2c_1c_2y(t_n) - c_1^{\,2}y(t_{n-1}) + \left[1 - c_1(c_2 + c_3)\right]x(t_n)$$

$$+ \left[c_1^{\,2} - c_1(c_2 - c_3)\right]x(t_{n-1}).$$

FORTRAN IV V01C-03A FRI 30-JUL-76 02:54:56 PAGE 001

```
      SUBROUTINE LPF
      COMMON T
      COMMON/CRESLT/RESULT(100),RPAST1,RPAST2
      COMMON/CDATA/JCTR,DATA(50)
      COMMON/CDATIN/DATAIN(100),DPAST1,DPAST2
      NPTS=100
      CALL COPY
      IF(DATA(JCTR).EQ.1) GO TO 10
      X=(6.283185*DATA(JCTR+1)*T)/SQRT(2.)
      C1=EXP(-X)
      C2=COS(X)
      C3=SIN(X)
      RESULT(1)=YNT(C1,C2,C3,RPAST1,RPAST2,DPAST1,DPAST2)
      RESULT(2)=YNT(C1,C2,C3,RESULT(1),RPAST1,DATAIN(1),DPAST)
      DO 20 J=3,NPTS
   20 RESULT(J)=YNT(C1,C2,C3,RESULT(J-1),RESULT(J-2),DATAIN(J-1),
     +DATAIN(J-2))
   40 RPAST1=RESULT(NPTS)
      RPAST2=RESULT(NPTS-1)
      DPAST1=DATAIN(NPTS)
      DPAST2=DATAIN(NPTS-1)
      RETURN
   10 EAT=EXP(-6.283185*DATA(JCTR+1)*T)
      RESULT(1)=EAT*RPAST1+(1-EAT)*DPAST1
      DO 30 K=2,100
   30 RESULT(K)=EAT*RESULT(K-1)+(1-EAT)*DATAIN(K-1)
      GO TO 40
      END
```

26. Subroutine MODIFY

Calls: None

Called by: SBCTRL

Commons: blank, CDATA, CCIRKT

MODIFY is the processor for changing the characteristics of a system configured from function blocks. It contains the text information for identifying the parameters associated with each available function block. The variable IBLOCK points to an entry in array ITYP where the block identifier and data location point are obtained. MODIFY uses this information to print the name of the block and list its parameters from the data array DATA. Modifications are made to the parameters in array DATA as desired by the user. The sets of RETURN-CONTINUE statements allow for later addition of other listings as new function elements are implemented.

```
0001          SUBROUTINE MODIFY(IBLOCK)
0002          COMMON T,J,IBLK,IQCNTR,IGRAFX,ICOM,ICNTR
0003          COMMON/CDATA/JCTR,DATA(50)
0004          COMMON/CCIRKT/NBLK,ITYP(10,2)
0005          DATA X/1H /,YES/1HY/
0006          ICOM=7
0007          IF(IBLOCK.GT.NBLK) GO TO 900
0009          IBTYP=ITYP(IBLOCK,1)
0010          LCTR=ITYP(IBLOCK,2)
0011          GO TO(10,20,30,40,50,60,70,80,90,100,110)IBTYP
0012       10 WRITE(6,12) IBLOCK
0013       12 FORMAT(' BLOCK ',I2,' IS A SIGNAL GENERATOR')
0014       11 JTYP=DATA(LCTR)
0015          JSMP=DATA(LCTR+4)
0016          WRITE(6,14)JTYP,DATA(LCTR+1),DATA(LCTR+2),DATA(LCTR+3),JSMP
0017       14 FORMAT(' 1. TYPE=',I2,' ( 0=SINE, 1=SQUARE WAVE )',
         +/' 2. AMPLITUDE= ',E10.4,' VOLTS',/' 3. FREQUENCY= ',E10.4,' HZ',
         +/' 4. DC OFFSET= ',E10.4,' VOLTS',/' 5. SAMPLES PER PERIOD=',I4)
0018          GO TO 700
0019       20 WRITE(6,22)IBLOCK
0020       22 FORMAT(' BLOCK ',I2,' IS A SAMPLE AND HOLD')
0021       21 JTYP=DATA(LCTR)
0022          JSMP=DATA(LCTR+1)
0023          WRITE(6,24)JTYP,JSMP
0024       24 FORMAT(/' 1. OUTPUT PULSE DURATION = ',I10,'% OF SAMPLING PERIOD',
         +/' 2. FIRST ORDER LPF WITH',4X,I10,' SECTIONS ( 0=NO FILTERING )')
0025          IF(JSMP.EQ.0) GO TO 700
0027          WRITE(6,25) DATA(LCTR+2)
0028       25 FORMAT(' 3. BLOCK BANDWIDTH',7X,'= ',E10.4,' HZ')
0029          GO TO 700
0030       30 WRITE(6,32) IBLOCK
0031       32 FORMAT(' BLOCK ',I2,' IS A QUANTIZER')
0032       31 NQL=DATA(LCTR+2)
0033          IQSMP=DATA(LCTR+3)
0034          IOUTSW=DATA(LCTR+4)
0035          IDOUT=DATA(LCTR+5)
0036          JTYP=DATA(LCTR+6)
0037          WRITE(6,34)DATA(LCTR),DATA(LCTR+1),
         +NQL,IQSMP,IOUTSW,IDOUT,JTYP
0038       34 FORMAT(/' 1.,2. DYNAMIC RANGE IS ',F5.2,' TO ',F5.2,' VOLTS.',
         +/'     3. NO. OF QUANTIZATION LEVELS  = ',I3,
         +/'     4. INPUT LATCHED IN AT ',I3,'% OF SAMPLE PERIOD',
         +/'     5. BINARY OUTPUT SWING IS ',I2,'   ( -1=SYMMETRIC',
         +', 0=NON-SYMMETRIC )',
         +/'     6. OUTPUT BIT DURATION IS ',I3,'% OF  BIT INTERVAL',
         +/'     7. NO. OF LPF SECTIONS = ',I2,' ( 0=NO FILTERING )')
0039          IF (JTYP.EQ.0) GO TO 700
0041          WRITE(6,36) DATA(LCTR+7)
0042       36 FORMAT('     8. BLOCK BANDWIDTH  ',E10.4)
0043          GO TO 700
0044       40 RETURN
0045       41 CONTINUE
0046       50 RETURN
0047       51 CONTINUE
```

```
   60 RETURN
   61 CONTINUE
   70 RETURN
   71 CONTINUE
   80 RETURN
   81 CONTINUE
   90 RETURN
   91 CONTINUE
  100 WRITE(6,102)IBLOCK
  102 FORMAT(' BLOCK ',I2,' IS A LOW PASS FILTER')
  101 JTYP=DATA(LCTR)
      WRITE(6,104)JTYP,DATA(LCTR+1)
  104 FORMAT(' 1. TYPE=',I2,' ( 1=RC LPF, 2=2ND ORDER BUTTERWORTH )',
     +/' 2. CORNER FREQUENCY= ',E10.4,' HZ')
      GO TO 700
  110 RETURN
  111 CONTINUE
  700 WRITE(6,710)
  710 FORMAT(' DO YOU WANT TO MAKE CHANGES?')
      READ(5,715)X
  715 FORMAT(A1)
      IF (X.NE.YES) RETURN
  717 READ(5,555)JPNTR,VALUE
  555 FORMAT(I10,F12.3)
      IF(JPNTR.EQ.0) GO TO 716
      DATA(LCTR+JPNTR-1)=VALUE
      GO TO 717
  716 GO TO (11,21,31,41,51,61,71,81,91,101,111)IBTYP
  900 WRITE(6,910)NBLK
  910 FORMAT(' ERROR...ONLY ',I2,' BLOCKS IN SYSTEM MODEL')
      RETURN
      END
```

27. Subroutines PE, PE70

Calls: ERF, CMQF

Called by: SY60, SY70

Commons: None

PE and PE70 are identical routines which compute probability-of-error vs. signal or energy-to-noise ratios for the six types of systems in the old version system level analysis package. The existence of both routines is a carry-over from the original two-region overlay structure designed for implementing the demonstration in a PDP-11. The following formulas were used for the analysis:

(1) coherent ASK: $$P_e = \frac{P_m}{2}\left[1 + \mathrm{erf}((\alpha - 1)\sqrt{ENR/2})\right] + \frac{1 - P_m}{2}\left[1 - \mathrm{erf}(\alpha\sqrt{ENR/2})\right],$$

(2) non-coherent ASK: $$P_e = (1 - P_m)e^{-y^2/2} + P_m Q_c(x,y),$$

(3) coherent PSK: $$P_e = \frac{P_m}{2}\left[1 - \mathrm{erf}((1 - \alpha)\sqrt{ENR/2})\right] + \frac{1 - P_m}{2}\left[1 - \mathrm{erf}((1 + \alpha)\sqrt{ENR/2})\right]$$

(4) non-coherent PSK: $$P_e = \frac{1}{2}e^{-SNR},$$

(5) coherent FSK: $$P_e = \frac{P_m}{2}\left[1 - \mathrm{erf}((2 - \alpha)\sqrt{ENR/2})\right] + \frac{1 - P_m}{2}\left[1 - \mathrm{erf}\left(\alpha\frac{\sqrt{ENR}}{2}\right)\right],$$

(6) non-coherent FSK: $P_e = \frac{1}{2} e^{-SNR/2}$,

where P_e = probability of error

P_m = probability of mark or space

erf = error function

ENR = energy-to-noise ratio

SNR = signal-to-noise ratio

y = normalized decision threshold

α = decision threshold

$Q_c(x,y)$ = complementary Marcum Q-function

$x = \sqrt{2 \cdot SNR}$

```
FORTRAN IV       V01C-03A   FRI 30-JUL-76 03:30:35                PAGE 001

      SUBROUTINE  PE(PERROR,I,K,A,ENR)
      DIMENSION A(10)
      TEMP=SQRT(ENR/2.)
      P=A(1)
      D=A(9)
      GO TO(10,20,30,40,50,60)I
   10 PERROR=(P*ERF((D-1)*TEMP)+(P-1)*ERF(D*TEMP)+1.)/2.
      RETURN
   20 PERROR=-(P*ERF((1-D)*TEMP)+(1-P)*ERF((1+D)*TEMP)-1.)/2.
      RETURN
   30 SQT=SQRT(2.)
      PERROR=-(P*ERF((2-D)*(TEMP/SQT))+(1-P)*ERF(D*(TEMP/SQT))-1)/2.
      RETURN
   40 SNR=-ENR/2.
      IF(ABS(SNR).GT.676.)GO TO 41
      PERROR=0.5*EXP(SNR)
      RETURN
   41 PERROR=0.
      RETURN
   50 SNR=-ENR
      IF(ABS(SNR).GT.676.) GO TO 41
      PERROR=0.5*EXP(SNR)
      RETURN
   60 Y=A(10)**2/2
      X=SQRT(2*ENR)
      IF(ABS(Y).GT.676.) GO TO 61
      PERROR=(1-P)*EXP(-Y)+P*CMQF(X,A(10))
      RETURN
   61 PERROR=P*CMQF(X,A(10))
      RETURN
      END
```

FORTRAN IV V01C-03A FRI 30-JUL-76 03:30:52 PAGE 001

```
      SUBROUTINE  PE70(PERROR,I,K,A,ENR)
      DIMENSION A(10)
      TEMP=SQRT(ENR/2.)
      P=A(1)
      D=A(9)
      GO TO(10,20,30,40,50,60)I
   10 PERROR=(P*ERF((D-1)*TEMP)+(P-1)*ERF(D*TEMP)+1.)/2.
      RETURN
   20 PERROR=-(P*ERF((1-D)*TEMP)+(1-P)*ERF((1+D)*TEMP)-1.)/2.
      RETURN
   30 SQT=SQRT(2.)
      PERROR=-(P*ERF((2-D)*(TEMP/SQT))+(1-P)*ERF(D*(TEMP/SQT))-1)/2.
      RETURN
   40 SNR=-ENR/2.
      IF(ABS(SNR).GT.676.)GO TO 41
      PERROR=0.5*EXP(SNR)
      RETURN
   41 PERROR=0.
      RETURN
   50 SNR=-ENR
      IF(ABS(SNR).GT.676.) GO TO 41
      PERROR=0.5*EXP(SNR)
      RETURN
   60 Y=A(10)**2/2
      X=SQRT(2*ENR)
      IF(ABS(Y).GT.676.) GO TO 61
      PERROR=(1-P)*EXP(-Y)+P*CMQF(X,A(10))
      RETURN
   61 PERROR=P*CMQF(X,A(10))
      RETURN
      END
```

28. Subroutine PICT1

Calls: FORTRAN graphics library

Called by: ASSIST

Commons: None

PICT1 generates a light pen sensitive list from which the user may choose the type of analysis he wants to pursue. The subpicture tag ITST is returned to the calling program to identify the choice and set the control variable ICOM accordingly.

*LST14

FORTRAN IV   V01C-03A   FRI 30-JUL-76 02:52:20   PAGE 001

```
      SUBROUTINE PICT1(ITST)
      DIMENSION IBUF(500)
      CALL INIT(IBUF,500)
D     CALL TIME(900)
D   1 CALL TIMR(I)
D     IF(I.NE.0) GO TO 1
      PAUSE 'TRY RE OR RSU TO CONT'
      CALL SCROL(1,1000,1)
      CALL SUBP(1)
      CALL APNT(200.,500.,1,-6,1)
      CALL TEXT('SYSTEM')
      CALL ESUB
      CALL SUBP(2)
      CALL APNT(600.,500.,1,-6,1)
      CALL TEXT('SUBFUNCTION')
      CALL ESUB
      CALL APNT(0.,100.,0,-5,-1)
      CALL TEXT('26-JUL-76',1,'*NOTE*',1,'SUBFUNCTION LEVEL SHOULD',
     +          ' BE USED TO ACCESS OLD VERSION SYSTEM LEVEL')
   10 CALL LPEN(M,ITST)
      IF(M.NE.1) GO TO 10
      IF(ITST.EQ.0) GO TO 20
      CALL ERAS(1)
      CALL ERAS(2)
      CALL SCROL(-1,-1,-1)
      CALL FREE
      RETURN
   20 M=0
      GO TO 10
      END
```

29. Subroutine PLTEND

Calls: None

Called by: GTPLT

Commons: blank

PLTEND is used to wrap up the plotting routine in GTPLT and reset the control variables ICOM and ICNTRL. These operations are done in this subroutine rather than in GTPLT in order to reduce memory size requirements for the overlay structure.

FORTRAN IV V01C-03A FRI 30-JUL-76 03:33:24 PAGE 001

```
      SUBROUTINE PLTEND
      COMMON T,J,IB,IQ,IG,ICOM,ICNTRL
      WRITE(6,1)
    1 FORMAT(/' END OF PLOT')
      ICOM=2
      ICNTRL=2
      PAUSE ' TYPE RE TO CONTINUE'
      RETURN
      END
```

30. Subroutine PRAB

Calls: ERF, SNR

Called by: SYSAST

Commons: CUPB, CYC (CUPB is not used)

PRAB is the analysis routine for the three types of ASK systems available in the new version system level package. The control variable M directs program flow to the appropriate sections for the system under consideration. This routine generates an array of probability-of-error data which is later used to produce bit-error-rate curves on the CRT.

(1) Non-coherent ASK: $P_e = (1 - P_m)e^{-y^2/2} + P_m Q_c(x,y)$

(2) Coherent ASK:
$$P_e = P_m\left[1 - \left(\frac{1}{2}\right)\mathrm{erfc}\left(\frac{b_o}{\sqrt{2}} - \sqrt{SNR}\right)\right] + (1 - P_m)\left[\frac{1}{2}\,\mathrm{erfc}\left(\frac{b_o}{\sqrt{2}}\right)\right]$$

(3) Correlation receiver ASK:
$$P_e = \frac{P_m}{2}\left[1 + \mathrm{erf}((\alpha - 1)\sqrt{ECR})\right] + \frac{1 - P_m}{2}\left[1 - \mathrm{erf}(\alpha\sqrt{ECR})\right]$$

where P_e = probability of error

P_m = probability of mark

y = normalized decision threshold

Q_c = complementary Marcum Q-function

b_o = normalized decision threshold

SNR = signal to noise ratio

$x = \sqrt{2 \cdot SNR}$

erfc = complementary error function

α = decision threshold

ECR = energy contrast ratio.

```
      SUBROUTINE PRAB(M)
      COMMON/CUPB/UPB(40),IBND
      COMMON/CYC/YC(40)
      DO 1 I=1,40
    1 YC(I)=0.0
      WRITE(6,100)
  100 FORMAT(' WHAT IS THE MARK PROBABILITY?')
      READ(5,101) PM
  101 FORMAT(F10.3)
      SQRT2=SQRT(2.)
      IF(M.EQ.3) GO TO 30
      WRITE(6,103)
  103 FORMAT(' WHAT IS THE NORMALIZED DECISION THRESHOLD?')
      READ(5,101) Y
      IF(M.EQ.2) GO TO 20
      DO 11 L=1,40
      YC(L)=(1.-PM)*EXP(-(Y**2.)/2.)+PM*CMQF(SQRT(2.*SNR(L)),Y)
      IF(YC(L).LT.1E-6) RETURN
   11 CONTINUE
      RETURN
   20 DO 21 L=1,40
      YC(L)=PM*(1-(.5)*ERF((Y/SQRT2)-SQRT(SNR(L))))
     *  +(1-PM)*(.5*ERF(Y/SQRT2))
      IF(YC(L).LT.1E-6) RETURN
   21 CONTINUE
      RETURN
   30 WRITE(6,104)
  104 FORMAT(' WHAT IS THE DECISION THRESHOLD? (.5 IS NORMAL)')
      READ(5,101) Y
      DO 31 L=1,40
      YC(L)=(PM/2)*(1.+ERF((Y-1.)*SQRT(SNR(L))))
     *  +((1-PM)/2)*(1-ERF(Y*SQRT(SNR(L))))
      IF(YC(L).LT.1E-6) RETURN
   31 CONTINUE
      RETURN
      END
```

31. Subroutine PRCTRL

Calls: PROCES

Called by: MAIN

Commons: HSTORY, blank, CDATA, CCIRKT, CWORD, CDATIN, CRESLT, CBER, CBINRY

PRCTRL is the controller which directs the simulation of a system configured from subfunction blocks. It maintains status information in control variables while processing blocks of time-sampled waveform data through the configured system. Its operation is flow charted in Figure A-1.

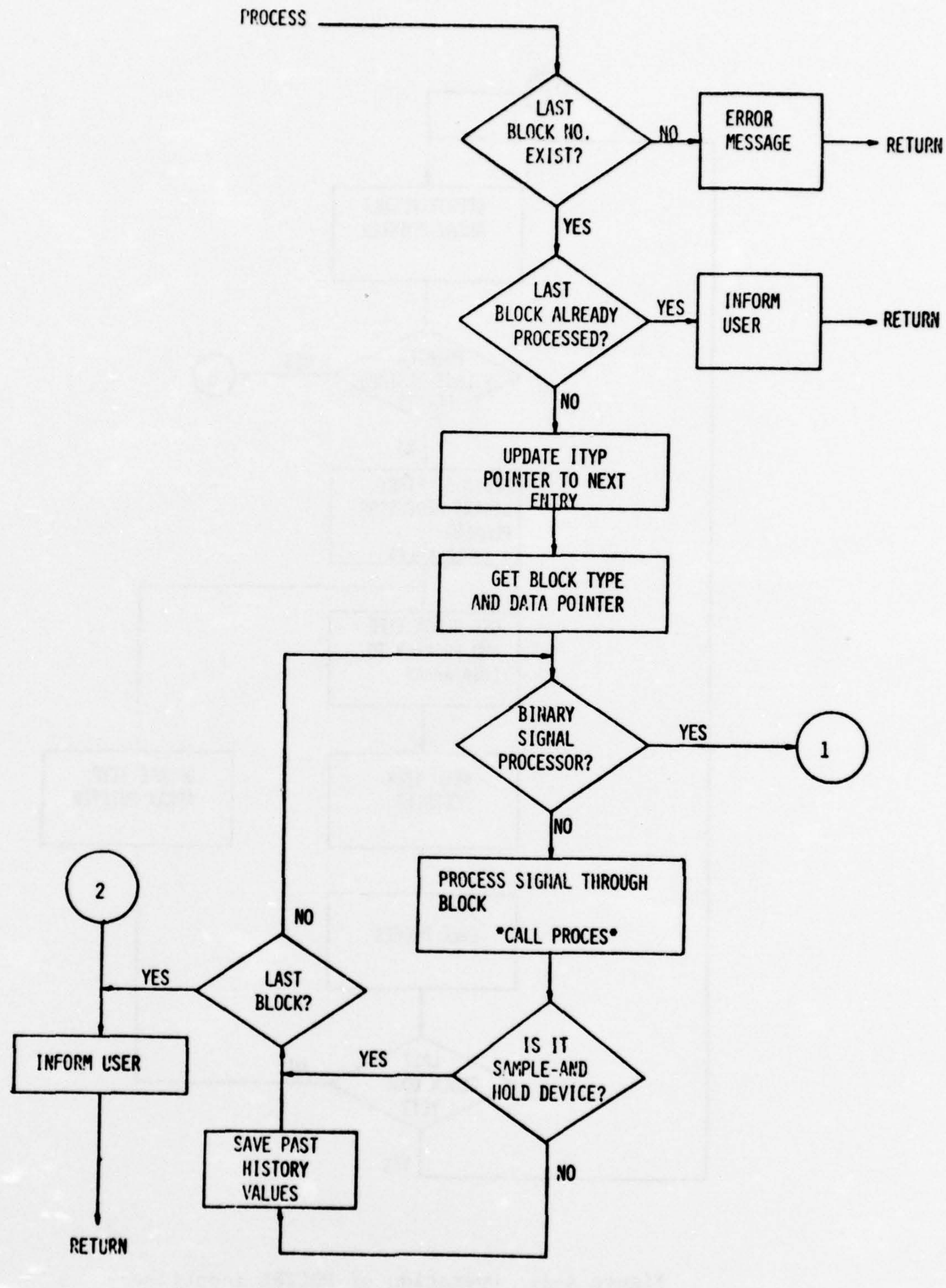


Figure A-1. Operation of PRCTRL

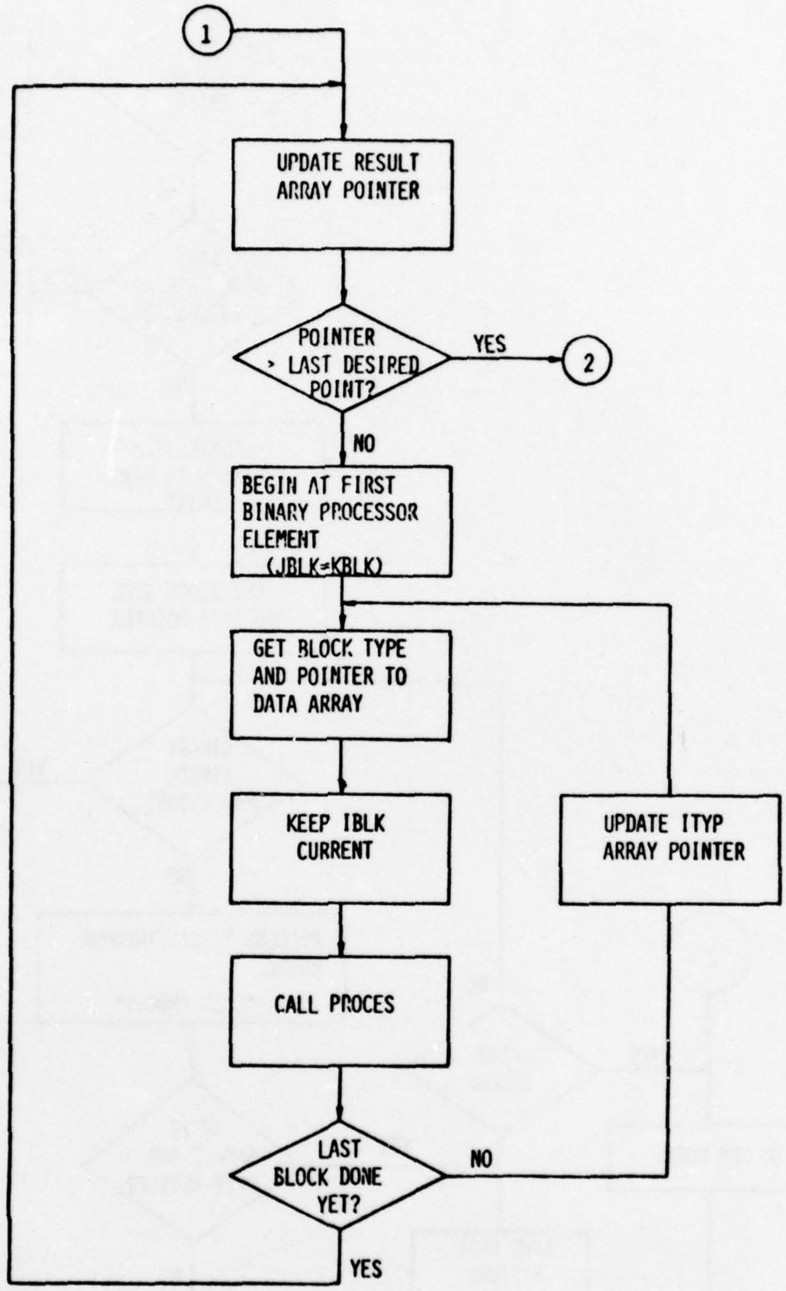


Figure A-1. Operation of PRCTRL (continued)

```
*LST
FORTRAN IV      V01C-03A    FRI 30-JUL-76 02:13:18                PAGE 001

      SUBROUTINE PRCTRL
      COMMON/HSTORY/A(10,4)
      COMMON T,J,IBLK,IQCNTR,IGRAFX,ICOM,ICNTRL,IPNTS
      COMMON/CDATA/JCTR,DATA(50)
      COMMON/CCIRKT/NBLK,ITYP(10,2)
      COMMON/CWORD/WORD(10)
      COMMON/CDATIN/DATAIN(100),DPAST1,DPAST2
      COMMON/CRESLT/RESULT(100),RPAST1,RPAST2
      COMMON/CBER/BER(40)/CBINRY/BINARY(100),IQWORD(5)
  120 NOUT=WORD(2)
      IF(NOUT.LE.NBLK) GO TO 291
      WRITE(6,7001)NBLK
   10 ICOM=7
      RETURN
  291 IF(NOUT-IBLK)295,295,293
 293  ITMP=IBLK+1
      DO 292 KBLK=ITMP,NOUT
      IBTYP=ITYP(KBLK,1)
      JCTR=ITYP(KBLK,2)
      IBLK=KBLK
      IF(IBTYP.GE.3.AND.IBTYP.LE.8) GO TO 296
      CALL PROCES(IBTYP)
      IF(IBTYP.EQ.2) GO TO 292
      A(KBLK,1)=RESULT(NPTS-1)
      A(KBLK,2)=RESULT(NPTS)
      A(KBLK,3)=DATAIN(NPTS-1)
      A(KBLK,4)=DATAIN(NPTS)
  292 CONTINUE
  295 WRITE(6,7002) IBLK
      GO TO 10
  296 IQCNTR=IQCNTR+1
      IF(IQCNTR.GT.IPNTS) GO TO 295
      DO 294 JBLK=KBLK,NOUT
      IBTYP=ITYP(JBLK,1)
      JCTR=ITYP(JBLK,2)
      IBLK=JBLK
  294 CALL PROCES(IBTYP)
      GO TO 296
 7001 FORMAT(' ERROR....THERE ARE ONLY ',I2,' BLOCKS IN THE SYSTEM')
 7002 FORMAT(' PROCESSING COMPLETE THROUGH BLOCK',I3)
      RETURN
      END
```

32. Subroutine PRFS

Calls: ERF, SNR

Called by: SYSAST

Commons: CYC

PRFS is the analysis routine for the two types of FSK systems available in the new version system level package. Variable M directs the program flow to the appropriate sequence. The data is saved in array YC for later conversion and plotting. The formulas used are given below:

(1) Non-coherent FSK: $P_e = \frac{1}{2} e^{-SNR/2}$

(2) Matched filter FSK: $$P_e = \frac{P_m}{2}\left[1 - \mathrm{erf}((2 - \alpha)\sqrt{ECR/2})\right] + \frac{1 - P_m}{2}\left[1 - \mathrm{erf}(\alpha\sqrt{ECR/2})\right]$$

where P_e = probability of error

erf = error function

α = decision threshold

ECR = energy contrast ratio

SNR = signal-to-noise ratio

P_m = probability of mark.

```
      SUBROUTINE PRFS(M)
      COMMON/CYC/YC(40)
      SQR(L)=SQRT(SNR(L)/2.)
      DO 1 I=1,40
    1 YC(I)=0.
      IF(M.EQ.9) GO TO 90
      DO 80 I=1,40
      YC(I)=.5*EXP(-SNR(I)/2.)
      IF(YC(I).LT.1.E-6) RETURN
   80 CONTINUE
      RETURN
   90 WRITE(6,100)
      READ(5,101) PM
      WRITE(6,102)
      READ(5,101) Y
      DO 91 I=1,40
      YC(I)=(PM/2.)*(1.-ERF((2.-Y)*SQR(I)))
     *  +((1-PM)/2.)*(1.-ERF(Y*SQR(I)))
      IF(YC(I).LT.1.E-6) RETURN
   91 CONTINUE
      RETURN
  100 FORMAT(' WHAT IS THE MARK PROBABILITY? (NORMAL IS 0.5)')
  101 FORMAT(F10.3)
  102 FORMAT(' WHAT IS THE DECISION THRESHOLD? (NORMAL IS 1.0)')
      END
```

33. Subroutine PROCES

Calls: SIGGEN, SMPHLD, QNTIZER, LPF, other unwritten routines

Called by: PRCTRL

Commons: None

PROCES is a switching routine that calls the proper function simulation routine for processing the signal.

```
      SUBROUTINE PROCES(I)
      GO TO ( 1,2,3,4,5,6,7,8,9,10,11),I
    1 CALL SIGGEN
      RETURN
    2 CALL SMPHLD
      RETURN
    3 CALL QNTIZER
      RETURN
    4 CALL NCODER
      RETURN
    5 CALL XMTR
      RETURN
    6 CALL CHANNL
      RETURN
    7 CALL RCVR
      RETURN
    8 CALL DECDEV
      RETURN
    9 CALL DCODER
      RETURN
   10 CALL LPF
      RETURN
   11 CALL BPF
      RETURN
      END
```

34. Subroutine PRPSD

Calls: ERF, SNR

Called by: SYSAST

Commons: CYC

PRPSD is the analysis routine for the three types of binary PSK systems available in the new system level package. Variable M directs program flow to the appropriate sequence. Both the coherent PSK and the correlation receiver PSK systems use the same formula as shown below. The data is saved in array YC for later conversion and plotting.

(1) Coherent PSK and correlation receiver PSK:

$$P_e = \frac{P_m}{2}\left[1 - \mathrm{erf}((1 - \alpha)\sqrt{ECR})\right] + \frac{1 - P_m}{2}\left[1 - \mathrm{erf}((1 + \alpha)\sqrt{ECR})\right]$$

(2) DPSK: $P_e = \frac{1}{2} e^{-SNR}$

where P_e = probability of error

P_m = probability of mark

erf = error function

ECR = energy contrast ratio (signal-to-noise ratio for coherent case)

SNR = signal-to-noise ratio

α = decision threshold.

```
FORTRAN IV        V01C-03A    FRI 30-JUL-76 03:26:52                PAGE 001

      SUBROUTINE PRPSD(M)
      COMMON/CYC/YC(40)
      DO 1 I=1,40
    1 YC(I)=0.
      IF(M.EQ.6) GO TO 60
      WRITE(6,100)
      READ(5,101) PS
      WRITE(6,102)
      READ(5,101) Y
  100 FORMAT(' WHAT IS THE MARK PROBABILITY? (NORMAL IS .5)')
  101 FORMAT(F10.3)
  102 FORMAT(' WHAT IS THE DECISION THRESHOLD? (NORMAL IS 0.0)')
      DO 41 I=1,40
      YC(I)=(PS/2.)*(1.-ERF((1+Y)*SQRT(SNR(I))))
     *  +((1-PS)/2.)*(1.-ERF((1-Y)*SQRT(SNR(I))))
      IF(YC(I).LT.1E-6) RETURN
   41 CONTINUE
      RETURN
   60 DO 61 I=1,40
      YC(I)=.5*EXP(-SNR(I))
      IF(YC(I).LT.1E-6) RETURN
   61 CONTINUE
      RETURN
      END
```

35. Subroutine PRQ

Calls: SNRF, SNR, ERF, R3YNO

Called by: SYSAST

Commons: CYC, CUPB

PRQ is the analysis routine for the four-phase PSK system included in the new system level package. Variable M is not used in this routine even though it is in the argument list. This routine optionally generates a second set of data for upper-bound plotting and stores it in array UPB. The QPSK data is stored in array YC. If the optional plot is desired, a control variable is set to inform the calling program upon return. The formulas used are shown below.

(1) QPSK: $P_e = \text{erfc}(\sqrt{SNR/2}) \left[1 - \frac{1}{4} \text{erfc}(\sqrt{SNR/2})\right]$

(2) Upper bound: $P_e \leq \text{erfc}\left(\sqrt{SNR} \sin\left(\frac{\pi}{m}\right)\right)$

where P_e = probability of error

erfc = complementary error function

SNR = signal-to-noise ratio

m = no. of phase states.

```
      SUBROUTINE PRQ(M)
      COMMON/CYC/YC(40)
      COMMON/CUPB/UPB(40),IBND
      DATA HY/1HY/
      ERFC(Z)=1-ERF(Z)
      SNRF(L)=SQRT(SNR(L)/2.)
      DO 1 I=1,40
      YC(I)=0.
    1 UPB(I)=0.
      PI=3.14159
      WRITE(6,100)
      CALL R3YNO(KK)
      IF(KK.NE.10) GO TO 71
      WRITE(6,102)
      READ(5,103) NSTATE
   71 DO 72 I=1,40
      YC(I)=ERFC(SNRF(I))*(1.-.25*ERFC(SNRF(I)))
      IF(YC(I).LT.1.E-6) GO TO 73
   72 CONTINUE
   73 IF(KK.NE.10) RETURN
      IBND=1
      DO 74 I=1,40
      UPB(I)=ERFC((SQRT(SNR(I)))*SIN(PI/FLOAT(NSTATE)))
      IF(UPB(I).LT.1.E-6) RETURN
   74 CONTINUE
      RETURN
  100 FORMAT(' DO YOU WANT TO COMPARE RESULTS WITH UPPER BOUND PLOT',
     +  ' FOR M-ARY SYSTEMS?')
  101 FORMAT(A1)
  102 FORMAT(' HOW MANY PHASE STATES?')
  103 FORMAT(I10)
      END
```

36. Subroutine QNTIZER

Calls: None

Called by: PROCES

Commons: CRESLT, CBINRY, CDATA, blank

QNTIZER is a function simulation for what might be more properly called a PCM encoder. It generates a serial binary output waveform representing the m-bit quantization of the input sample amplitude, where m ranges from 1 to 5. The output waveform is stored in array BINARY as 5 bits with 20 time sampled values per bit, optionally bandlimited. The routine uses a successive approximation technique to convert the input amplitude into a binary word. QNTIZER writes the idealized values of the output to the terminal device while the actual time waveform is generated as described above.

```
*LST20
FORTRAN IV       V01C-03A   FRI 30-JUL-76 02:55:26              PAGE 001

      SUBROUTINE QNTIZER
      COMMON/CRESLT/RESULT(100)
      COMMON/CBINRY/BINARY(100),IQWORD(5)
      COMMON/CDATA/JCTR,DATA(50)
      COMMON T,JJJ,IBLK,IQCNTR
      DIMENSION A(20)
      MIN=DATA(JCTR)
      RG=DATA(JCTR+1)-MIN
      NUMBIT=IFIX(ALOG(DATA(JCTR+2))/ALOG(2.)+.99)
      DATA(JCTR+2)=2**NUMBIT
      ISECT=DATA(JCTR+6)
      IF(ISECT.EQ.0) GO TO 10
      BLKBW=DATA(JCTR+7)
      EAT=FEAT1(ISECT,BLKBW)
   10 IBPT=11-DATA(JCTR+5)/10.
      IEPT=10+DATA(JCTR+5)/10.
      ICNT=0
      MIN1=MIN
      DO 45 J=1,NUMBIT
      TST=MIN1+RG/2**J
      IF(RESULT(IQCNTR).GT.TST) GO TO 20
      DO 11 I=1,20
      A(I)=0.
   11 IF(IBPT.LE.I.AND.I.LE.IEPT) A(I)=DATA(JCTR+4)
      IQWORD(J)=DATA(JCTR+4)
      GO TO 30
   20 DO 21 I=1,20
      A(I)=0.
   21 IF(IBPT.LE.I.AND.I.LE.IEPT) A(I)=1.
      IQWORD(J)=1
      MIN1=TST
   30 IF(ISECT.EQ.0) GO TO 40
      DO 35 I=1,ISECT
   35 CALL SAHLPF(EAT,A)
   40 DO 45 I=1,20
      ICNT=ICNT+1
   45 BINARY(ICNT)=A(I)
      WRITE(6,100)(IQWORD(I),I=1,NUMBIT)
  100 FORMAT(5I3)
      RETURN
      END
```

37. Subroutine R3YNO

Calls: FORTRAN Graphics Library

Called by: SYSAST, SYSINT

Commons: None

R3YNO sets up a region 3 display buffer and restores a light-pen-sensitive 'YES-NO' list to the CRT screen. When a hit occurs, the sub-picture tag corresponding to a 'YES' or a 'NO' is returned to the calling program through the variable I.

```
FORTRAN IV      V01C-03A   FRI 30-JUL-76 03:33:50                    PAGE 001

0001            SUBROUTINE R3YNO(I)
0002            DIMENSION IBUF(100)
0003            CALL INIT(IBUF,100)
0004            CALL SCROL(5,900,5)
0005            CALL RSTR('YNO')
0006         10 CALL LPEN(M,I)
0007            IF(M.EQ.0) GO TO 10
0009            IF(I.EQ.0) GO TO 20
0011            CALL FREE
0012            CALL SCROL(-1,-1,-1)
0013            RETURN
0014         20 M=0
0015            GO TO 10
0016            END
```

38. Subroutines SAHLPF, SMPLPF

Calls: None

Called by: SMPHLD (SMPLPF), QNTIZER (SAHLPF)

Commons: blank, HSTORY

SAHLPF and SMPLPF are identical routines which are used to simulate band-limited characteristics for the elements being modeled in the calling programs. Each performs a copying operation to obtain an input array, retrieves past history information from array C in common block HSTORY, processes the input array through a single section RC low pass filter, updates the past history array. Depending on user-supplied parameters, the calling program calls SAHLPF or SMPLPF one, two, or three times in succession to simulate a one, two, or three section filtering characteristic. The formula is the same as used in subroutine LPF for the simple RC case. The duplication of routines was required in the original 2-level overlay structure but causes no problems in the existing 3-level structure.

```
      SUBROUTINE SAHLPF(EAT,A)
      COMMON/HSTORY/C(10,4)
      COMMON T,I,IBLK
      DIMENSION A(20),B(20)
      DO 10 K=1,20
   10 B(K)=A(K)
      SMPR1=C(IBLK,2)
      SMPD1=C(IBLK,4)
      A(1)=EAT*SMPR1+(1-EAT)*SMPD1
      DO 20 K=2,20
   20 A(K)=EAT*A(K-1)+(1-EAT)*B(K-1)
      C(IBLK,4)=B(20)
      C(IBLK,2)=A(20)
      RETURN
      END
```

```
      SUBROUTINE SMPLPF(EAT,A)
      COMMON/HSTORY/C(10,4)
      COMMON T,I,IBLK
      DIMENSION A(20),B(20)
      DO 10 K=1,20
   10 B(K)=A(K)
      SMPR1=C(IBLK,2)
      SMPD1=C(IBLK,4)
      A(1)=EAT*SMPR1+(1-EAT)*SMPD1
      DO 20 K=2,20
   20 A(K)=EAT*A(K-1)+(1-EAT)*B(K-1)
      C(IBLK,4)=B(20)
      C(IBLK,2)=A(20)
      RETURN
      END
```

39. Subroutine SBCTRL

Calls: DATIT, FETCH, ELFIND, STRDTA, GOOF, MODIFY, TIMPLT

Called by: MAIN

Commons: HSTORY, blank, CDATA, CCIRKT, CWORD, CDATIN, CRESLT, CBER CBINRY

SBCTRL is the main control program for the subfunction level analysis package. It prompts the user for input with a '*' character and a line feed. Then the command input and decoding routines are called, and the returned control variable LTYP is used to direct the program flow with the computed GO TO statement. Since PDP-11 FORTRAN does not allow entry points to be set directly in subroutines, the control variable ICNTRL and the logical IF statement are used to achieve the same effect. Re-entry at statement label 1 occurs after a normal exit from either of the two system level analysis packages. The large number of calls to GOOF and the CONTINUE statements allow for later addition of subroutine calls or other operations as the software is developed, without extensive changes to the existing programs.

```
*LST2
FORTRAN IV        V01C-03A   FRI 30-JUL-76 02:13:52                    PAGE 001

          SUBROUTINE SBCTRL(ISYSTM)
          COMMON/HSTORY/A(10,4)
          COMMON T,J,IBLK,IQCNTR,IGRAFX,ICOM,ICNTRL,IPNTS
          COMMON/CDATA/JCTR,DATA(50)
          COMMON/CCIRKT/NBLK,ITYP(10,2)
          COMMON/CWORD/WORD(10)
          COMMON/CDATIN/DATAIN(100),DPAST1,DPAST2
          COMMON/CRESLT/RESULT(100),RPAST1,RPAST2
          COMMON/CBER/BER(40)/CBINRY/BINARY(100),IQWORD(5)
          IF (ICNTRL.EQ.1) GO TO 1
      200 NBLK=0
      210 DO 203 LCNT=1,10
          DO 203 JCNT=1,4
      203 A(LCNT,JCNT)=0.0
          WRITE(6,8300)
     8305 FORMAT(I4)
     8300 FORMAT(' SPECIFY NO. OF POINTS TO PROCESS THRU QUANTIZER')
          READ(5,8305) IPNTS
          J=1
      220 IBLK=0
          IQCNTR=0
          JCTR=1
          ITYP(1,2)=1
        1 CALL DATIT
          WRITE(6,7010)
     7010 FORMAT(2H *)
          CALL FETCH (L,NBAD)
          IF (NBAD.EQ.0) GO TO 1
          CALL ELFIND (WORD,LTYP)
          GO TO ( 10, 20, 30, 40, 50, 60, 70, 80, 90,100,
         +        110,120,130,140,150,160,170,180,190,200,
         +        210,220,230,240,250,260,270,280,290,300,
         +        310,320,330,340,350,360,370,380,380,380,
         +        380,380,380,380,380,460),LTYP
       10 CALL STRDTA(5)
          NTYP=1
          GO TO 600
       20 CALL STRDTA(3)
          NTYP=2
          GO TO 600
       30 CALL STRDTA(8)
          NTYP=3
          GO TO 600
       40 CALL GOOF
          GO TO 1
       50 CALL GOOF
          GO TO 1
       60 CALL GOOF
          GO TO 1
       70 CALL GOOF
          GO TO 1
       80 CALL GOOF
          GO TO 1
       90 CALL GOOF
```

```
      GO TO 1
  100 CALL STRDTA(2)
      NTYP=10
      GO TO 600
  110 CALL GOOF
      GO TO 1
  120 ICOM=5
      RETURN
  130 JBLK=WORD(2)
      CALL MODIFY(JBLK)
      GO TO 1
  140 CALL GOOF
      GO TO 1
  150 CALL TIMPLT
      GO TO 1
  160 CALL GOOF
      GO TO 1
  170 ICOM=14
      RETURN
  180 CALL GOOF
      GO TO 1
  190 CALL GOOF
      RETURN
  230 ISYSTM=1
  231 ICOM=2
      RETURN
  240 ISYSTM=2
      GO TO 231
  250 ISYSTM=3
      GO TO 231
  260 ISYSTM=4
      GO TO 231
  270 ISYSTM=5
      GO TO 231
  280 ISYSTM=6
      GO TO 231
  290 ICOM=10
      RETURN
  300 CONTINUE
  310 CONTINUE
  320 CONTINUE
  330 CONTINUE
  340 CONTINUE
  350 CONTINUE
  360 CONTINUE
  370 CONTINUE
  380 WRITE(6,381)
  381 FORMAT(' INVALID COMMAND IN SUBFUNCTION ANALYSIS')
      GO TO 1
  460 WRITE(6,7003)
 7003 FORMAT ('  UNDEFINED STATEMENT  ')
      GO TO 1
  600 NBLK=NBLK+1
      ITYP(NBLK,1)=NTYP
```

```
      ITYP (NBLK+1,2)=JCTR
      GO TO 1
      END
```

40. Subroutine SIGGEN

Calls: None

Called by: PROCES

Commons: CRESLT, CDATA, blank

SIGGEN is the signal generator element for the function-level analysis package. It generates either sine or square waves with characteristics specified by the user. The time sampled waveform output is stored in array RESULT. The control variable J keeps track of the location of the last sample relative to the beginning of the period.

The data is generated in blocks of 100 samples at a time, and the variable J is used to remember where the last sample of the previous block was located so the generator can create a continuous signal without transient effects between sets of data. The formulas shown below were used.

(1) Square wave: $x(t) = \begin{cases} \text{amplitude} + \text{DC offset, 1st half period} \\ -\text{amplitude} + \text{DC offset, 2nd half period} \end{cases}$

(2) Sine wave: $x(jT) = A\sin(2\pi fjT) + \text{DC offset}$

where A = amplitude (volts)

F = frequency (Hz)

$T = (F \cdot (\text{samples/period}))^{-1}$

j = current sample pointer.

```
*LST17
FORTRAN IV        V01C-03A   FRI 30-JUL-76 02:54:41                  PAGE 001

            SUBROUTINE SIGGEN
            COMMON/CRESLT/RESULT(100)
            COMMON/CDATA/JCTR,DATA(50)
            COMMON T,J
            AMP=DATA(JCTR+1)
            FREQ=DATA(JCTR+2)
            DCOFST=DATA(JCTR+3)
            ISMPPD=DATA(JCTR+4)
            T=1/(FREQ*ISMPPD)
            W=2.*3.141927*FREQ*T
            IHALF=DATA(JCTR+4)/2.
            IF(DATA(JCTR).EQ.0) GO TO 10
            DO 20 K=1,100
            IF(J.LE.IHALF) RESULT(K)=AMP+DCOFST
            IF(J.GT.IHALF) RESULT(K)=-AMP+DCOFST
            J=J+1
            IF(J.GT.ISMPPD) J=1
         20 CONTINUE
            RETURN
         10 DO 30 K=1,100
            RESULT(K)=AMP*SIN(J*W)+DCOFST
            J=J+1
            IF(J.GT.ISMPPD) J=1
         30 CONTINUE
            RETURN
            END
```

41. Subroutine SLST

Calls: FORTRAN Graphics Library

Called by: SYSAST

Commons: None

SLST sets up a region 2 display buffer and restores a light pen sensitive list of nine types of communications systems available for analysis in the new system level package. It then waits for a light pen hit and returns the sub-picture tag N of the hit to the calling program.

```
*R2SLST
FORTRAN IV        V01C-03A   FRI 30-JUL-76 03:18:45                    PAGE 001

0001           SUBROUTINE SLST(N)
0002           DIMENSION IBUF(200)
0003           CALL INIT(IBUF,200)
0004           CALL SCROL(5,900,5)
0005           CALL RSTR('SLST')
0006        10 CALL LPEN(M,N)
0007           IF(M.EQ.0) GO TO 10
0009           IF(N.EQ.0) GO TO 20
0011           CALL FREE
0012           RETURN
0013        20 M=0
0014           GO TO 10
0015           END
```

42. Subroutine SMPHLD

Calls: HOLD, SMPLPF

Called by: PROCES

Commons: CDATA, CCIRKT, CRESLT, HSTORY, blank

SMPHLD simulates a sample-and-hold device, optionally with band-limited characteristics. It does a look-ahead in the ITYP array to determine if it is followed by a valid element, in this case the quantizer. If so, it saves only the particular output sample needed for input to the next element. If not, it writes the entire set of generated sample points to a disk file named FTN12.DAT. The operation of SMPHLD is detailed in the flow chart, Figure A-2. The program listing shows the lines with a 'D' in column 1 as being ignored. This occurred in the listing only. The final compiled program includes the extra lines.

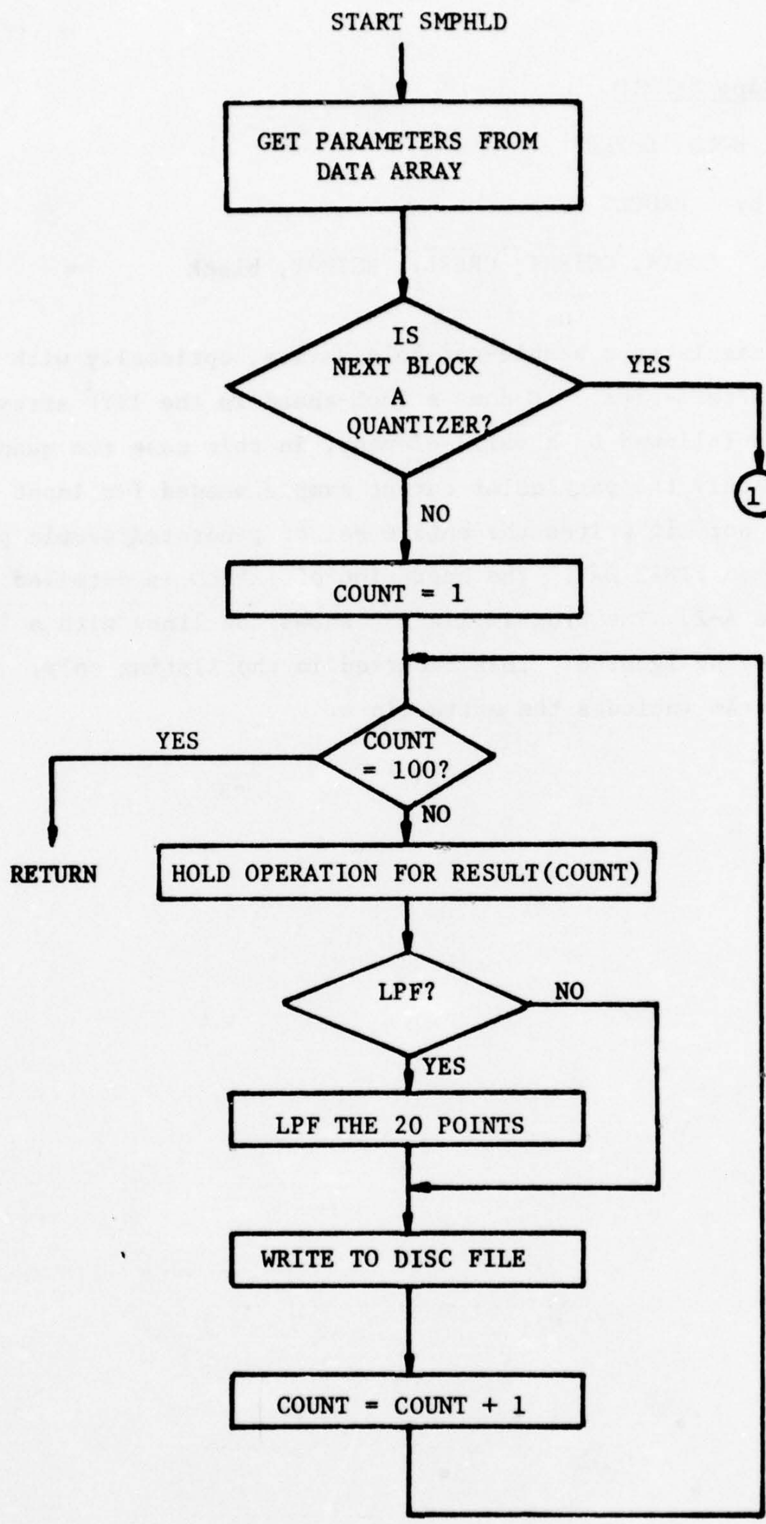


Figure A-2. Operation of Subroutine SMPHLD

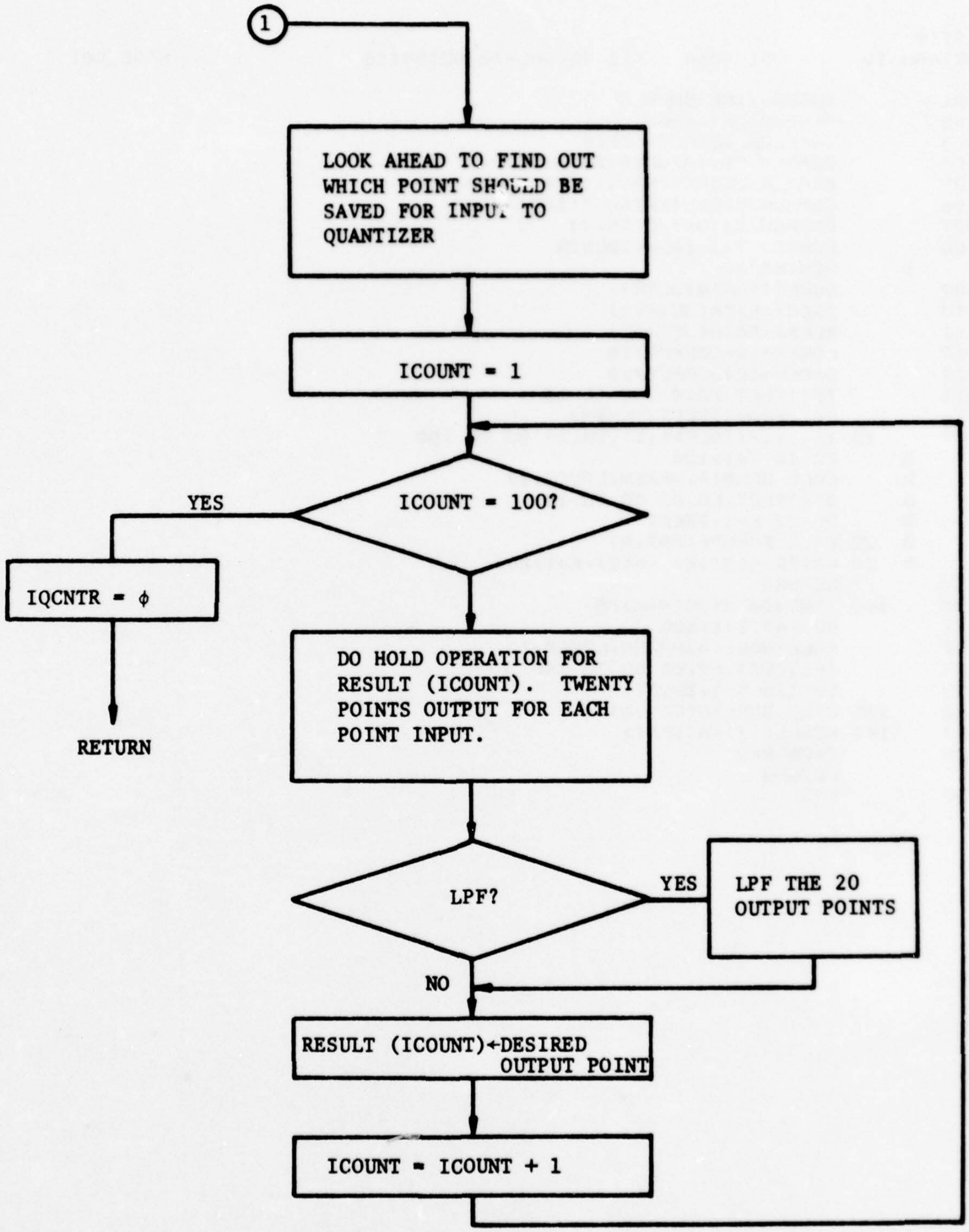


Figure A-2. Operation of Subroutine SMPHLD (continued).

```
*LST19
FORTRAN IV       V01C-03A   FRI 30-JUL-76 02:55:10                    PAGE 001

0001          SUBROUTINE SMPHLD
0002          DIMENSION A(20)
0003          INTEGER DURPCT,UPPER
0004          COMMON/CDATA/JCTR,DATA(50)
0005          COMMON/CCIRKT/NBLK,ITYP(10,2)
0006          COMMON/CRESLT/RESULT(100)
0007          COMMON/HSTORY/C(10,4)
0008          COMMON T,J,IBLK,IQCNTR
       D      REWIND 12
0009          DURPCT=DATA(JCTR)
0010          ISECT=DATA(JCTR+1)
0011          BLKBW=DATA(JCTR+2)
0012          LOWER=10-DURPCT/10
0013          UPPER=10+DURPCT/10
0014          IF(ISECT.EQ.0) GO TO 15
0016          EAT=FEAT(ISECT,BLKBW)
0017       15 IF(ITYP(IBLK+1,1).EQ.3) GO TO 100
       D      DO 10 I=1,100
       D      CALL HOLD(A,UPPER,LOWER,I)
       D      IF(ISECT.EQ.0) GO TO 10
       D      DO 22 K=1,ISECT
       D   22 CALL SMPLPF(EAT,A)
       D   10 WRITE (12'12) (A(K),K=1,20)
0019          RETURN
0020      100 IPNT=DATA(JCTR+6)/5
0021          DO 140 I=1,100
0022          CALL HOLD(A,UPPER,LOWER,I)
0023          IF(ISECT.EQ.0) GO TO 140
0025          DO 135 K=1,ISECT
0026      135 CALL SMPLPF(EAT,A)
0027      140 RESULT(I)=A(IPNT)
0028          IQCNTR=0
0029          RETURN
0030          END
```

43. Function SNR

Calls: None

Called by: PRAB, PRFS, PRPSD, PRQ

Commons: None

SNR computes a value of signal-to-noise ratio which depends on the value of the argument L. For L from 1 to 41, SNR generates values which range in dB from -10 to 30. The formula is given below:

$$SNR = 10^{\frac{L-11}{10}}$$

```
FORTRAN IV        V01C-03A   FRI 30-JUL-76 03:33:08                PAGE 001

      FUNCTION SNR(L)
      SNR=10.**(FLOAT(L-11)/10.)
      RETURN
      END
```

44. Function SNRF

Calls: AMP, ETA

Called by: SYSTEM, SY60

Commons: None

SNRF evaluates and returns the following expression:

$$SNRF = A^2/2N,$$

where A = amplitude of received sinusoid

N = noise power

$= \eta B_n/2$

$\eta = 2 \cdot$ noise spectral density

B_n = noise bandwidth of receiver in Hz.

Note that some of these calculations take place in AMP and ETA. The result is a linearized (not in dB) value of signal-to-noise ratio.

```
FORTRAN IV      V01C-03A   FRI 30-JUL-76 03:30:37                PAGE 001

      FUNCTION SNRF(A,K)
      DIMENSION A(10)
      BN=A(9)
      PN=ETA(A,K)*BN/2.
      SNRF=AMP(A)**2/(2.*PN)
      RETURN
      END
```

45. Subroutine STRDTA

Calls: None

Called by: SBCTRL

Commons: CDATA, CCIRKT, CWORD

STRDTA takes parameters for function blocks from array WORD (placed there by FETCH) and stores them in the sequential data storage array DATA beginning at the location specified by JCTR. This operation is basic to the concept of configuring a system from a set of predefined blocks having variable parameters associated with them. The arrays ITYP and DATA together define the structure of the configured system.

```
FORTRAN IV      V01C-03A   FRI 30-JUL-76 02:31:30                PAGE 001

      SUBROUTINE STRDTA(K1)
      COMMON/CDATA/ JCTR,DATA(50)
      COMMON /CCIRKT/ NBLK,ITYP(10,2)
      COMMON /CWORD/ WORD(10)
      JCTR=ITYP(NBLK+ 1,2)
      DO 10 I=1,K1
   10 DATA(JCTR+I-1)=WORD(I+1)
      JCTR=JCTR+K1
  999 RETURN
      END
```

46. Subroutine SYSAST

Calls: SYSINT, SLST, AB1, AB2, AB3, PS1, PS2, DPS, QPS, FS1, FS2, SYSMSS, R3YNO, PRAB, PRPSD, PRFS, PRQ, CNVT, DSPSNR, DSPECR

Called by: MAIN

Commons: CYC, blank, CUPB

SYSAST is the overall control routine for the new version system level package. It primarily serves as a switching routine to call other routines as directed by control variables M and I, which in turn depend on user responses to inquiries. M primarily identifies which system is under consideration, while I identifies YES-NO responses which affect program flow.

```
      SUBROUTINE SYSAST
      COMMON/CYC/YC(40)
      COMMON T,J,IB,IQ,IG,ICOM,ICNTRL
      COMMON/CUFB/UFB(40),IBND
      DO 1 I=1,40
    1 UFB(I)=0.
      CALL SYSINT
D     PAUSE 'TYPE RE OR RSU TO CONTINUE'
  100 CALL SLST(M)
      GO TO (10,20,30,40,50,60,70,80,90) M
   10 CALL AB1
      GO TO 110
   20 CALL AB2
      GO TO 110
   30 CALL AB3
      GO TO 110
   40 CALL PS1
      GO TO 110
   50 CALL PS2
      GO TO 110
   60 CALL DPS
      GO TO 110
   70 CALL QPS
      GO TO 110
   80 CALL FS1
      GO TO 110
   90 CALL FS2
  110 CALL SYSMSS(1)
      CALL R3YNO(I)
      IF(I.EQ.11) GO TO 120
      GO TO 100
  120 IF(M.EQ.7) GO TO 155
      IF(M.GT.7) GO TO 150
      IF(M.GE.4) GO TO 140
      CALL PRAB(M)
      GO TO 160
  140 CALL PRPSD(M)
      GO TO 160
  150 CALL PRFS(M)
      GO TO 160
  155 CALL PRQ(M)
  160 CALL CNVT
      IF((M.EQ.3).OR.(M.EQ.5).OR.(M.EQ.9)) GO TO 170
      CALL DSPSNR
      GO TO 180
  170 CALL DSPECR
  180 CALL SYSMSS(2)
      CALL R3YNO(I)
      IF(I.EQ.10) GO TO 120
      CALL SYSMSS(3)
      CALL R3YNO(I)
      IF(I.EQ.10) GO TO 100
      ICOM=1
      ICNTRL=1
```

```
0062        RETURN
0063        END
```

47. Subroutine SYSINT

Calls: R3YNO

Called by: SYSAST

Commons: None

SYSINT contains the text introduction to the new version system level package. It provides the option of skipping the introduction if desired by using R3YNO in conjunction with the appropriate inquiry.

```
*LST21
FORTRAN IV       V01C-03A   FRI 30-JUL-76 03:09:55                PAGE 001

0001          SUBROUTINE SYSINT
0002          WRITE(6,11)
0003       11 FORMAT(' DO YOU WANT TO READ INTRODUCTION? (USE LP)')
0004          CALL R3YNO(I)
0005          IF(I.EQ.11) GO TO 20
0007          WRITE(6,10)
0008       10 FORMAT('      THIS IS AN ANALYSIS PACKAGE FOR NINE',/,' TYPES ',
             *  ' OF COMMUNICATIONS SYSTEMS.  YOU WILL',/,' BE SHOWN A ',
             *  ' LIST OF THE SYSTEMS AVAILABLE AND',/,' YOU SHOULD USE ',
             *  ' THE LIGHT PEN TO SELECT THE SYSTEM',/,' OF INTEREST.',//,
             *  '      WHEN DATA IS SOLICITED, PROVIDE NUMBERS IN REAL',/,
             *  ' FORMAT (INCLUDE DECIMAL POINT OR USE EXPONENTIAL ',
             *  ' NOTATION) USING',/,' THE KEYBOARD.',//,'      THIS ',
             *  ' PACKAGE CALCULATES AND PLOTS THE BIT ERROR',/,' RATE ',
             *  ' VS.  SIGNAL OR ENERGY TO NOISE RATIO FOR THE SYSTEM',/,
             *  ' OF INTEREST.')
0009       20 RETURN
0010          END
```

48. Subroutine SYSMSS

Calls: None

Called by: SYSAST

Commons: None

SYSMSS contains the text data for all inquiries in SYSAST. The control variable I directs program flow to the desired sequence. This subroutine reduces the memory requirements for SYSAST in overlay region 1 by relocating text data in overlay region 2.

```
      SUBROUTINE SYSMSS(I)
      IF(I.EQ.3) GO TO 30
      IF(I.EQ.2) GO TO 20
      WRITE(6,11)
      RETURN
   20 WRITE(6,21)
      RETURN
   30 WRITE(6,31)
      RETURN
   11 FORMAT(' YOU MAY AT THIS TIME SELECT ANOTHER SYSTEM.',
     +/' DO YOU WANT TO DO SO? (USE LIGHT PEN)')
   21 FORMAT(' DO YOU WANT TO ALTER THE SYSTEM PARAMETERS?')
   31 FORMAT(' DO YOU WANT TO LOOK AT ANOTHER SYSTEM?')
      END
```

49. Subroutine SYSTEM

Calls: HINT, SY1MS, DATIT, FETCH, ELFIND, SY30, SY40, SY50, SY60, SY70, SNRF

Called by: MAIN

Commons: HDATA, DF1012, blank, CBER, CWORD, CSYSTM

SYSTEM is the overall control routine for the old version system level analysis package. It operates in a fashion similar to SBCTRL, utilizing the command decoding processors and a computed GO TO statement to direct program flow. The variable I identifies the system under consideration. An entry point is set at statement label 554 for re-entry after indirectly calling GTPLT, since both routines are assigned to the same overlay region.

```
          SUBROUTINE SYSTEM(I)
          COMMON/HDATA/HH(4),HX,HZ(9)
          COMMON/DF1012/IREC,NREC
          COMMON T,J,IBLK,IQ,IGRAFX,ICOM,ICNTRL
          COMMON/CBER/BER(40)
          COMMON/CWORD/WORD(10)
          COMMON/CSYSTM/SDATA(10),KTYP
          DATA HT/1HT/
          DIMENSION A(10)
          IF(ICNTRL.EQ.2) GO TO 554
          IGTST=0
       18 CALL HINT(I,JTYP,IGTST,K)
          CALL SY1MS(HX,HH(4),1)
      554 ICOM=7
        1 CALL DATIT
          WRITE(6,1001)
     1001 FORMAT(' /')
          CALL FETCH(L,NBAD)
          IF(NBAD.EQ.0) GO TO 1
          CALL ELFIND(WORD,LTYP)
          GO TO (10,10,10,10,10,10,10,10,10,10,
         +       10,60,10,70,10,10,10,10,10,10,
         +       10,10,15,16,17,12,13,14,10,10,
         +       10,10,10,10,10,10,10,20,30,40,
         +       50,60,10,80,90,95),LTYP
       10 CALL SY1MS(HX,HH(4),2)
          GO TO 1
       12 I=4
          GO TO 18
       13 I=5
          GO TO 18
       14 I=6
          GO TO 18
       15 I=1
          GO TO 18
       16 I=2
          GO TO 18
       17 I=3
          GO TO 18
D      20 REWIND 10
D         READ(10'1)(A(J),J=1,10),K
D         GO TO 1
       30 IGTST=1
          CALL HINT(I,JTYP,IGTST,K)
          CALL SY30(I,K,A)
          IGTST=0
          GO TO 555
       40 CALL SY40(A,K)
          GO TO 555
       50 CALL SY50(A,K)
      555 IF(I.NE.6) GO TO 1
          GO TO 200
       60 CALL SY60(JTYP,I,A,K)
          GO TO 1
```

```
0054          70 CALL SY70(IGTST,I,K,A)
0055             IF(ICOM.EQ.12) RETURN
0057             GO TO 1
0058          80 RETURN
0059     D    90 REWIND 10
0060     D       WRITE(10'1) (A(J),J=1,10),K
0061     D       GO TO 1
0062          95 CALL SY1MS(HX,HH(4),3)
0063             GO TO 1
0064         200 XXX=SQRT(2+SNRF(A,K)/2)
0065             CALL SY1MS(XXX,HH(4),4)
0066             IF(LTYP.EQ.40) A(10)=XXX
0068             GO TO 1
***** E
```

50. Subroutine SY1MS

Calls: None

Called by: SYSTEM

Commons: None

SY1MS handles the text messages for SYSTEM in a fashion similar to SYSMSS. This reduces memory requirements for overlay region 1. A and B contain Hollerith variable text which is generated by HINT before SY1MS is called.

```
*LST16
FORTRAN IV        V01C-03A    FRI 30-JUL-76 02:54:29              PAGE 001

      SUBROUTINE SY1MS(A,B,ISM)
      GO TO (10,20,30,40) ISM
   10 WRITE(6,1000) A,B
 1000 FORMAT(//10X,' SYSTEM LEVEL ANALYSIS FOR BINARY ',A2,'SK',
     +/' ASSUMPTIONS: 1. ',A4,'COHERENT DETECTION',/13X,
     +' 2. ADDITIVE WHITE GAUSSIAN NOISE')
      RETURN
   20 WRITE(6,1002)
 1002 FORMAT(' INVALID... USE SYSTEM LEVEL ANALYSIS COMMANDS')
      RETURN
   30 WRITE(6,96)
   96 FORMAT(' UNDEFINED STATEMENT')
      RETURN
   40 WRITE(6,201) A
  201 FORMAT(' CURRENT ESTIMATE OF OPTIMUM THRESHOLD IS ',F8.2)
      RETURN
      END
```

51. Subroutine SY30

Calls: None

Called by: SYSTEM

Commons: HDATA

SY30 performs the parameter listing operation in which the current system parameters are listed on the terminal in response to the 'LIST' command. Hollerith variable data is inserted in the text output at appropriate places.

```
FORTRAN IV        V01C-03A   FRI 30-JUL-76 03:30:09            PAGE 001

0001          SUBROUTINE SY30(I,K,A)
0002          COMMON/HDATA/HH(4),HX,HZ(9)
0003          DIMENSION A(10)
0004       31 WRITE(6,32) HX,(A(J),J=1,7),HH(1),HH(2),A(8),HH(3),
             +(HZ(M),M=1,5),A(9),HZ(6)
0005       32 FORMAT(//'      SYSTEM IS'19X,A2,'SK',
             +/' 1. MARK PROBABILITY',5X,F10.2,
             +/' 2. MESSAGE RATE',9X,E10.4,' /SEC',
             +/' 3. XMTR POWER',11X,E10.4,' WATTS',
             +/' 4. XMTR ANT. GAIN',7X,F10.2,' DB',
             +/' 5. FREQUENCY',12X,E10.4,' HZ',/' 6. RCVR-XMTR SEPARATION ',
             +E10.4,' METERS',/' 7. RCVR ANT. GAIN',7X,F10.2,' DB',
             +/' 8. NOISE ',A4,A2,9X,F10.2,1X,A4,/' 9. ',4A4,A2,
             +3X,F10.2,A3)
0006          IF(I.NE.6) GO TO 1
0008          WRITE(6,33) A(10)
0009       33 FORMAT(' 10. NORMALIZED DECISION THRESHOLD = ',F8.2)
0010       35 WRITE(6,34)
0011       34 FORMAT(/)
0012        1 RETURN
0013          END
```

52. Subroutine SY40

Calls: None

Called by: SYSTEM

Commons: HDATA

SY40 is the parameter entry processor in which new system parameters are input via a user-computer dialogue. It is called in response to the 'NEW' command. The data array A contains the system parameters.

```
      SUBROUTINE SY40(A,K)
      DIMENSION A(10)
      COMMON/HDATA/HH(4),HX,HZ(9)
      WRITE(6,41)
   41 FORMAT(//' TRANSMITTER:',/' WHAT IS PROBABILITY OF A MARK?')
   42 READ(5,400) A(1)
  400 FORMAT(F10.3)
      IF(0..LE. A(1).AND. A(1).LE.1.) GO TO 110
      WRITE(6,1100)
 1100 FORMAT(' ENTRY ERROR...VALUE SHOULD BE BETWEEN 0 AND 1.')
      GO TO 42
  110 WRITE(6,120)
  120 FORMAT(' INPUT MESSAGE RATE/SEC, XMTR POWER,',
     +' XMTR ANT. GAIN (DB), FREQ. (HZ)')
      READ(5,410)(A(J),J=2,5)
  410 FORMAT(4E12.3)
      WRITE(6,130)
  130 FORMAT(/' CHANNEL:',/' INPUT XMTR-RCVR SEPARATION',
     +' DISTANCE (METERS)')
      READ(5,420) A(6)
  420 FORMAT(E12.3)
      WRITE(6,140)
  140 FORMAT(/' RECEIVER',/' WILL NOISE FIGURE BE IN (1) DB, OR',
     +' (-1) KELVIN DEGREES?')
   43 READ(5,430)K
  430 FORMAT(I4)
      IF(IABS(K).NE.1) GO TO 43
   45 WRITE(6,150) (HZ(M),M=1,5)
  150 FORMAT(' INPUT RCVR ANT. GAIN (DB), NOISE FIGURE, AND ',4A4,A2)
      A(10)=0.
      READ(5,440) A(7),A(8),A(9)
  440 FORMAT(3F12.3)
      RETURN
      END
```

53. Subroutine SY50

Calls: None

Called by: SYSTEM

Commons: CWORD

SY50 is called in response to the 'CHANGE' command. It looks at the data in the WORD array (placed there by FETCH) to find out which entry in data array A is to be changed and the new value to be stored in that entry. It also tests to see if the user has specified the toggling operation for array element A(8) and sets control variable K accordingly. A(8) is the noise specification and can be given either as a temperature or a noise figure.

```
FORTRAN IV        V01C-03A    FRI 30-JUL-76 03:30:13                PAGE 001

0001          SUBROUTINE SY50(A,K)
0002          DATA HT/1HT/
0003          DIMENSION A(10)
0004          COMMON/CWORD/WORD(10)
0005          KPAR=WORD(2)
0006          A(KPAR)=WORD(3)
0007          IF(WORD(4).EQ.HT.AND.KPAR.EQ.8) K=-K
0009          RETURN
0010          END
```

54. Subroutine SY60

Calls: ENRF, SNRF, PE, DRATEF

Called by: SYSTEM

Commons: HDATA

SY60 is the analysis routine which computes and outputs the data rate, signal or energy-to-noise ratio, and probability of error for the set of parameters in data array A, using a number of calls to functions and subroutines in which the mathematical operations take place. The energy or signal-to-noise ratio is converted to dB before being output to the terminal. SY60 is called in response to the 'PROCES' command.

```
FORTRAN IV        V01C-03A   FRI 30-JUL-76 03:30:32                   PAGE 001

0001          SUBROUTINE SY60(JTYP,I,A,K)
0002          DIMENSION A(10)
0003          COMMON/HDATA/HH(4),HX,HZ(9)
0004       60 ENR=ENRF(A,K)
0005          IF(JTYP.EQ.1) ENR=SNRF(A,K)
0007          BO=SQRT(2.+(ENR/2.))
0008          CALL PE(PERROR,I,K,A,ENR)
0009          ENR=10.*ALOG10(ENR)
0010          DRATE=DRATEF(A)
0011          WRITE(6,61) DRATE,HZ(8),HZ(9),ENR,PERROR
0012       61 FORMAT(//' DATA RATE',13X,2H= ,E10.4,' BITS/SEC',
             +/1X,A4,A2,' TO NOISE RATIO = ',F10.2,' DB',
             +/' PROBABILITY OF ERROR  = ',E10.4)
0013          RETURN
0014          END
```

55. Subroutine SY70

Calls: PE70

Called by: SYSTEM

Commons: blank, HDATA, CBER

SY70 is called in response to the 'BERPLT' command. It performs a parametric analysis of bit error rate vs. signal-to-noise ratio over a range from -10 to 30 dB. Only the mark probability and decision threshold data from data array A is used. The output is stored in array BER for conversion and plotting by GTPLT, or, if control variable IGRAFX indicates that no CRT display is being used, it lists the array contents on the terminal.

```
FORTRAN IV      V01C-03A    FRI 30-JUL-76 03:30:50                  PAGE 001

            SUBROUTINE SY70(IGTST,I,K,A)
            DIMENSION A(10)
            COMMON T,J,IBLK,IQCNTR,IGRAFX,ICOM,ICNTRL
            COMMON/HDATA/HH(4),HX,HZ(9)
            COMMON/CBER/BER(40)
         70 DO 71 L=1,40
         71 BER(L)=0.0
            DO 72 L=1,40
            SNR=10.**(FLOAT(L-11)/10.)
            CALL PE70(PERROR,I,K,A,SNR)
            LCNT=L
            IF(PERROR.LT.1.E-6) GO TO 74
         72 BER(L)=PERROR
         74 IF(IGRAFX.NE.1) GO TO 78
            ICOM=12
            ICNTRL=LCNT
            RETURN
         78 WRITE(6,73) HZ(7)
         73 FORMAT(11X,A1,'NR ( DB )',10X,'PROB. OF ERROR',/)
            DO 75 L=1,40
            IF(BER(L).LT.1.E-6) GO TO 76
            SNRDB=FLOAT(L-11)
         75 WRITE(6,77) SNRDB,BER(L)
         77 FORMAT(10X,F10.2,15X,E10.4)
         76 RETURN
            END
```

56. Subroutine TIMPLT

Calls: None

Called by: SBCTRL

Commons: blank, CRESLT

TIMPLT generates a teletype plot of the data in array RESULT. The routine provides the capability of plotting selected portions of the array by soliciting from the user the starting point in the data array, the ending point, and the number of data array points to be skipped between plotted points. The routine automatically scales the plot and gives the minimum and maximum values plotted. The axis labeling is not automatically scaled; it is fixed to range from 0 to 1.0. Interpolation must be used to determine actual values of points plotted.

```
        SUBROUTINE TIMPLT
        COMMON T,J,IBLK,IQCNTR,IGRAFX,ICOM,ICNTR
        COMMON/CRESLT/RESULT(100)
        INTEGER*2 AST,BLNK
        DATA AST/1H*/,BLNK/1H /
        DIMENSION IA(50)
        WRITE(6,1000)
        ICOM=7
        READ(5,555) NST,NSP,NJUMP
   555  FORMAT(3I10)
D       WRITE(6,2000) (RESULT(I),I=NST,NSP,NJUMP)
D2000   FORMAT(10X,E12.3)
        BMAX=RESULT(NST)
        BMIN=BMAX
        DO 1  I=NST,NSP,NJUMP
        B=RESULT(I)
        IF(B.LT.BMIN) BMIN=B
     1  IF(B.GT.BMAX) BMAX=B
        IF((BMAX-BMIN).LT.1E-30) GO TO 999
        DO 33 I=1,50
    33  IA(I)=BLNK
   100  WRITE(6,4)
        WRITE(6,1001)T
        WRITE(6,3) BMIN,BMAX
        WRITE(6,4)
        WRITE(6,6)
        WRITE(6,7)
        DO 10 I=NST,NSP,NJUMP
        B=RESULT(I)
        B=(B-BMIN)/(BMAX-BMIN)
        M=IFIX(B*50.+.5)
D       WRITE(6,9090)M
D9090   FORMAT(' M=',I4)
        IF(M.EQ.0) GOTO 34
        IA(M)=AST
        WRITE(6,35)I,IA
        IA(M)=BLNK
        GO TO 36
    34  WRITE(6,37)I
    36  CONTINUE
    10  CONTINUE
        WRITE(6,11)
        GO TO 900
   999  WRITE(6,9)
   900  RETURN
     3  FORMAT(12X,'AMPLITUDE: MIN ',E10.3,', MAX ',E10.3,' VOLTS')
     4  FORMAT(5X,1H+)
     6  FORMAT(8X,' 0    .1    .2    .3    .4    .5    .6    .7    .8    .9  1.0')
     7  FORMAT(9X,1HI,10(5H----I))
     9  FORMAT(1X,' ERROR IN TIMPLT')
    11  FORMAT(6X,1HN)
    35  FORMAT(2X,I5,3H  I,50A1)
    37  FORMAT(2X,I5,3H  *)
  1000  FORMAT(' ENTER START,STOP, AND JUMP INDICES')
```

```
0055    1001 FORMAT(///' TIMESTEP= ',E10.4)
0056         END
```

57. Function YNT

Calls: None

Called by: LPF

Commons: None

YNT evaluates the expression for the 2nd order Butterworth low-pass filter given in the description accompanying the listing for LPF, using the values of the arguments in the argument list.

```
      FUNCTION YNT(C1,C2,C3,YN1,YN2,XN1,XN2)
      YNT=2.*C1*C2*YN1-C1*C1*YN2+(1-C1*(C2+C3))*XN1+C1*(C1-C2+C3)*XN2
      RETURN
      END
```

APPENDIX B

PROGRAMS FOR GENERATING PERMANENT
DISPLAY FILES USED BY MIDACS
DEMONSTRATION PROGRAM

APPENDIX B

PROGRAMS FOR GENERATING PERMANENT DISPLAY FILES USED BY MIDACS DEMONSTRATION PROGRAM

Each of these programs was used to generate a permanent, disc-resident display file for use by the MIDACS software. These display files all have the .DPY extension name and constitute the block diagrams, light pen sensitive lists, and coordinate grids displayed on the CRT. The graphics support subroutines are part of the PDP-11 FORTRAN GT Graphics Library supplied with the RT-11 operating system.

```
*LIST2
FORTRAN IV        V01C-03A    WED 25-AUG-76 13:36:11                    PAGE 0C

        C   ASKB1
        C
0001            DIMENSION IBUF(1000)
0002            CALL INIT (IBUF,1000)
0003            CALL APNT (50.,700.,0,-5)
0004            CALL TEXT ('BLOCK DIAGRAM ')
        C
        C   BEGIN BLOCK 1
        C
0005            CALL APNT (150.,550.,0,-5)
0006            CALL VECT (150.,0.)
0007            CALL VECT (0.,100.)
0008            CALL VECT (-150.,0.)
0009            CALL VECT (0.,-100.)
        C
        C   BEGIN BLOCK 2
        C
0010            CALL APNT (350.,550.,0,-5)
0011            CALL VECT (150.,0.)
0012            CALL VECT (0.,100.)
0013            CALL VECT (-150.,0.)
0014            CALL VECT (0.,-100.)
        C
        C   BEGIN BLOCK 3
        C
0015            CALL APNT (550.,550.,0,-5)
0016            CALL VECT (150.,0.)
0017            CALL VECT (0.,100.)
0018            CALL VECT (-150.,0.)
0019            CALL VECT (0.,-100.)
        C
        C   BEGIN BLOCK 4
        C
0020            CALL APNT (750.,550.,0,-5)
0021            CALL VECT (150.,0.)
0022            CALL VECT (0.,100.)
0023            CALL VECT (-150.,0.)
0024            CALL VECT (0.,-100.)
        C
        C   CONNECT LINES
        C
0025            CALL APNT (50.,600.,0,-5)
0026            CALL VECT (100.,0.)
0027            CALL VECT (150.,0.,0,-5)
0028            CALL VECT (50.,0.)
0029            CALL VECT (150.,0.,0,-5)
0030            CALL VECT (50.,0.)
0031            CALL VECT (150.,0.,0,-5)
0032            CALL VECT (50.,0.)
0033            CALL VECT (150.,0.,0,-5)
0034            CALL VECT (100.,0.)
        C
        C   LABEL BLOCKS
        C
```

```
0035        CALL APNT (160.,625.,0,-5)
0036        CALL TEXT ('BANDPASS ')
0037        CALL APNT (160.,575.,0,-5)
0038        CALL TEXT ('FILTER ')
0039        CALL APNT (360.,625.,0,-5)
0040        CALL TEXT ('HALFWAVE ')
0041        CALL APNT (360.,575.,0,-5)
0042        CALL TEXT ('DETECTOR ')
0043        CALL APNT (560.,625.,0,-5)
0044        CALL TEXT ('LOWPASS ')
0045        CALL APNT (560.,575.,0,-5)
0046        CALL TEXT ('FILTER ')
0047        CALL APNT (760.,625.,0,-5)
0048        CALL TEXT ('SAMPLE ')
0049        CALL APNT (760.,600.,0,-5)
0050        CALL TEXT ('AND ')
0051        CALL APNT (760.,575.,0,-5)
0052        CALL TEXT ('DECISION ')
0053        CALL APNT (910.,605.,0,-5)
0054        CALL TEXT ('OUTPUT ')
0055        CALL APNT (50.,605.,0,-5)
0056        CALL TEXT ('INPUT ')
      C
      C  LIST ASSUMPTIONS
      C
0057        CALL APNT (50.,450.,0,-5)
0058        CALL TEXT ('ASSUMPTIONS: ')
0059        CALL TEXT (1,'1. BINARY ASK ',1)
0060        CALL TEXT ('2. NON-COHERENT DETECTION ',1)
0061        CALL TEXT ('3. ADDITIVE WHITE GAUSSIAN NOISE ',1)
0062        CALL TEXT ('4. DECISION THRESHOLD NORMALIZED WITH RESPECT ',
           1      'TO RMS NOISE LEVEL ',1)
0063        CALL TEXT ('5. SIGNAL TO NOISE RATIO MEASURED AT OUTPUT OF ',
           1      'BANDPASS FILTER ',1)
0064        CALL TEXT ('6. NO INTERSYMBOL INTERFERENCE ')
0065        CALL SAVE ('ASKB1')
0066        CALL INIT (IBUF,1000)
0067        CALL RSTR ('ASKB1')
0068        END
```

```
ADJUST PAPER AND PRESS <RETURN> KEY
FORTRAN IV      V01C-03A   WED 04-AUG-76 13:37:31                    PAGE 001

         C   ASKB2
         C
0001           DIMENSION IBUF(1000)
0002           CALL INIT (IBUF,1000)
0003           CALL APNT (50.,700.,0,-5)
0004           CALL TEXT ('BLOCK DIAGRAM ')
         C
         C   BEGIN BLOCK 1
         C
0005           CALL APNT (150.,550.,0,-5)
0006           CALL VECT (150.,0.)
0007           CALL VECT (0.,100.)
0008           CALL VECT (-150.,0.)
0009           CALL VECT (0.,-100.)
         C
         C   BEGIN BLOCK 2
         C
0010           CALL APNT (350.,550.,0,-5)
0011           CALL VECT (150.,0.)
0012           CALL VECT (0.,100.)
0013           CALL VECT (-150.,0.)
0014           CALL VECT (0.,-100.)
         C
         C   BEGIN BLOCK 3
         C
0015           CALL APNT (550.,550.,0,-5)
0016           CALL VECT (150.,0.)
0017           CALL VECT (0.,100.)
0018           CALL VECT (-150.,0.)
0019           CALL VECT (0.,-100.)
         C
         C   BEGIN BLOCK 4
         C
0020           CALL APNT (750.,550.,0,-5)
0021           CALL VECT (150.,0.)
0022           CALL VECT (0.,100.)
0023           CALL VECT (-150.,0.)
0024           CALL VECT (0.,-100.)
         C
         C   CONNECT LINES
         C
0025           CALL APNT (50.,600.,0,-5)
0026           CALL VECT (100.,0.)
0027           CALL VECT (150.,0.,0,-5)
0028           CALL VECT (50.,0.)
0029           CALL VECT (150.,0.,0,-5)
0030           CALL VECT (50.,0.)
0031           CALL VECT (150.,0.,0,-5)
0032           CALL VECT (50.,0.)
0033           CALL VECT (150.,0.,0,-5)
0034           CALL VECT (100.,0.)
         C
         C   LABEL BLOCKS
         C
```

```
      CALL APNT (160.,625.,0,-5)
      CALL TEXT ('BANDPASS ')
      CALL APNT (160.,575.,0,-5)
      CALL TEXT ('FILTER ')
      CALL APNT (360.,625.,0,-5)
      CALL TEXT ('PRODUCT ')
      CALL APNT (360.,575.,0,-5)
      CALL TEXT ('DETECTOR ')
      CALL APNT (560.,625.,0,-5)
      CALL TEXT ('LOWPASS ')
      CALL APNT (560.,575.,0,-5)
      CALL TEXT ('FILTER ')
      CALL APNT (760.,625.,0,-5)
      CALL TEXT ('SAMPLE ')
      CALL APNT (760.,600.,0,-5)
      CALL TEXT ('AND ')
      CALL APNT (760.,575.,0,-5)
      CALL TEXT ('DECISION ')
      CALL APNT (910.,605.,0,-5)
      CALL TEXT ('OUTPUT ')
      CALL APNT (50.,605.,0,-5)
      CALL TEXT ('INPUT ')
C
C  LIST ASSUMPTIONS
C
      CALL APNT (50.,450.,0,-5)
      CALL TEXT ('ASSUMPTIONS: ')
      CALL TEXT (1,'1. BINARY ASK ',1)
      CALL TEXT ('2. COHERENT DETECTION ',1)
      CALL TEXT ('3. ADDITIVE WHITE GAUSSIAN NOISE ',1)
      CALL TEXT ('4. DECISION THRESHOLD NORMALIZED WITH RESPECT ',
     1         'TO RMS NOISE LEVEL ',1)
      CALL TEXT ('5. SIGNAL TO NOISE RATIO MEASURED AT OUTPUT OF ',
     1         'BANDPASS FILTER ',1)
      CALL TEXT ('6. NO INTERSYMBOL INTERFERENCE ')
      CALL SAVE ('ASKB2T')
      CALL INIT (IBUF,1000)
      CALL RSTR ('ASKB2')
      END
```

```
      C  ASKB3
      C
0001        DIMENSION IBUF(1000)
0002        CALL INIT (IBUF,1000)
0003        CALL APNT (50.,700.,0,-5)
0004        CALL TEXT ('BLOCK DIAGRAM ')
      C
      C  BEGIN BLOCK 1
      C
0005        CALL APNT (150.,550.,0,-5)
0006        CALL VECT (150.,0.)
0007        CALL VECT (0.,100.)
0008        CALL VECT (-150.,0.)
0009        CALL VECT (0.,-100.)
      C
      C  BEGIN BLOCK 2
      C
0010        CALL APNT (350.,550.,0,-5)
0011        CALL VECT (150.,0.)
0012        CALL VECT (0.,100.)
0013        CALL VECT (-150.,0.)
0014        CALL VECT (0.,-100.)
      C
      C  BEGIN BLOCK 3
      C
0015        CALL APNT (550.,550.,0,-5)
0016        CALL VECT (150.,0.)
0017        CALL VECT (0.,100.)
0018        CALL VECT (-150.,0.)
0019        CALL VECT (0.,-100.)
      C
      C  CONNECT LINES
      C
0020        CALL APNT (50.,600.,0,-5)
0021        CALL VECT (100.,0.)
0022        CALL VECT (150.,0.,0,-5)
0023        CALL VECT (50.,0.)
0024        CALL VECT (150.,0.,0,-5)
0025        CALL VECT (50.,0.)
0026        CALL VECT (150.,0.,0,-5)
0027        CALL VECT (100.,0.)
      C
      C  LABEL BLOCKS
      C
0028        CALL APNT (160.,625.,0,-5)
0029        CALL TEXT ('PRODUCT ')
0030        CALL APNT (160.,575.,0,-5)
0031        CALL TEXT ('DETECTOR ')
0032        CALL APNT (360.,625.,0,-5)
0033        CALL TEXT ('INTEGRATE ')
0034        CALL APNT (360.,600.,0,-5)
0035        CALL TEXT ('AND ')
0036        CALL APNT (360.,575.,0,-5)
0037        CALL TEXT ('DUMP ')
```

```
0038         CALL APNT (560.,625.,0,-5)
0039         CALL TEXT ('SAMPLE ')
0040         CALL APNT (560.,600.,0,-5)
0041         CALL TEXT ('AND ')
0042         CALL APNT (560.,575.,0,-5)
0043         CALL TEXT ('DECISION ')
0044         CALL APNT (710.,605.,0,-5)
0045         CALL TEXT ('OUTPUT ')
0046         CALL APNT (50.,605.,0,-5)
0047         CALL TEXT ('INPUT ')
       C
       C  LIST ASSUMPTIONS
       C
0048         CALL APNT (50.,450.,0,-5)
0049         CALL TEXT ('ASSUMPTIONS: ')
0050         CALL TEXT (1,'1. BINARY ASK ',1)
0051         CALL TEXT ('2. DETECTION BY A CORRELATION RECEIVER ',1)
0052         CALL TEXT ('3. ADDITIVE WHITE GAUSSIAN NOISE ',1)
0053         CALL TEXT ('4. DECISION THRESHOLD NORMALIZED WITH RESPECT ',
            1      'TO SIGNAL ENERGY ',1)
0054         CALL TEXT ('5. ENERGY CONTRAST RATIO MEASURED AT OUTPUT OF ',
            1      'INTEGRATE AND DUMP ',1)
0055         CALL TEXT ('6. NO INTERSYMBOL INTERFERENCE ')
0056         CALL SAVE ('ASKB3')
0057         CALL INIT (IBUF,1000)
0058         CALL RSTR ('ASKB3')
0059         END
```

```
FORTRAN IV      V01C-03A   WED 04-AUG-76 13:37:40                PAGE 001

        C   FSKB1
        C
0001            DIMENSION IBUF(1000)
0002            CALL INIT (IBUF,1000)
0003            CALL APNT (50.,700.,0,-5)
0004            CALL TEXT ('BLOCK DIAGRAM ')
        C
        C   BEGIN BLOCK 1
        C
0005            CALL APNT (150.,550.,0,-5)
0006            CALL VECT (150.,0.)
0007            CALL VECT (0.,100.)
0008            CALL VECT (-150.,0.)
0009            CALL VECT (0.,-100.)
        C
        C   BEGIN BLOCK 2
        C
0010            CALL APNT (350.,550.,0,-5)
0011            CALL VECT (150.,0.)
0012            CALL VECT (0.,100.)
0013            CALL VECT (-150.,0.)
0014            CALL VECT (0.,-100.)
        C
        C   BEGIN BLOCK 3
        C
0015            CALL APNT (550.,550.,0,-5)
0016            CALL VECT (150.,0.)
0017            CALL VECT (0.,100.)
0018            CALL VECT (-150.,0.)
0019            CALL VECT (0.,-100.)
        C
        C   BEGIN BLOCK 4
        C
0020            CALL APNT (750.,550.,0,-5)
0021            CALL VECT (150.,0.)
0022            CALL VECT (0.,100.)
0023            CALL VECT (-150.,0.)
0024            CALL VECT (0.,-100.)
        C
        C   CONNECT LINES
        C
0025            CALL APNT (50.,600.,0,-5)
0026            CALL VECT (100.,0.)
0027            CALL VECT (150.,0.,0,-5)
0028            CALL VECT (50.,0.)
0029            CALL VECT (150.,0.,0,-5)
0030            CALL VECT (50.,0.)
0031            CALL VECT (150.,0.,0,-5)
0032            CALL VECT (50.,0.)
0033            CALL VECT (150.,0.,0,-5)
0034            CALL VECT (100.,0.)
        C
        C   LABEL BLOCKS
        C
```

```
      CALL APNT (160.,625.,0,-5)
      CALL TEXT ('BANDPASS ')
      CALL APNT (160.,575.,0,-5)
      CALL TEXT ('FILTER ')
      CALL APNT (360.,625.,0,-5)
      CALL TEXT ('PRODUCT ')
      CALL APNT (360.,575.,0,-5)
      CALL TEXT ('DETECTOR ')
      CALL APNT (560.,625.,0,-5)
      CALL TEXT ('LOWPASS ')
      CALL APNT (560.,575.,0,-5)
      CALL TEXT ('FILTER ')
      CALL APNT (760.,625.,0,-5)
      CALL TEXT ('SIGN ')
      CALL APNT (760.,575.,0,-5)
      CALL TEXT ('DETECTOR ')
      CALL APNT (910.,605.,0,-5)
      CALL TEXT ('OUTPUT ')
      CALL APNT (50.,605.,0,-5)
      CALL TEXT ('INPUT ')
C
C  LIST ASSUMPTIONS
C
      CALL APNT (50.,450.,0,-5)
      CALL TEXT ('ASSUMPTIONS: ')
      CALL TEXT (1,'1. BINARY PSK ',1)
      CALL TEXT ('2. COHERENT DETECTION ',1)
      CALL TEXT ('3. ADDITIVE WHITE GAUSSIAN NOISE ',1)
      CALL TEXT ('4. DECISION THRESHOLD NORMALIZED WITH RESPECT ',
     1     'TO MAGNITUDE OF SIGNAL ',1)
      CALL TEXT ('     ENVELOPE ',1)
      CALL TEXT ('5. SIGNAL TO NOISE RATIO MEASURED AT OUTPUT OF ',
     1     'BANDPASS FILTER ',1)
      CALL TEXT ('6. NO INTERSYMBOL INTERFERENCE ')
      CALL SAVE ('PSKB1')
      CALL INIT (IBUF,1000)
      CALL RSTR ('PSKB1')
      END
```

```
          C  PSKB2
          C
0001            DIMENSION IBUF(1000)
0002            CALL INIT (IBUF,1000)
0003            CALL APNT (50.,700.,0,-5)
0004            CALL TEXT ('BLOCK DIAGRAM ')
          C
          C  BEGIN BLOCK 1
          C
0005            CALL APNT (150.,550.,0,-5)
0006            CALL VECT (150.,0.)
0007            CALL VECT (0.,100.)
0008            CALL VECT (-150.,0.)
0009            CALL VECT (0.,-100.)
          C
          C  BEGIN BLOCK 2
          C
0010            CALL APNT (350.,550.,0,-5)
0011            CALL VECT (150.,0.)
0012            CALL VECT (0.,100.)
0013            CALL VECT (-150.,0.)
0014            CALL VECT (0.,-100.)
          C
          C  BEGIN BLOCK 3
          C
0015            CALL APNT (550.,550.,0,-5)
0016            CALL VECT (150.,0.)
0017            CALL VECT (0.,100.)
0018            CALL VECT (-150.,0.)
0019            CALL VECT (0.,-100.)
          C
          C  BEGIN BLOCK 4
          C
0020            CALL APNT (750.,550.,0,-5)
0021            CALL VECT (150.,0.)
0022            CALL VECT (0.,100.)
0023            CALL VECT (-150.,0.)
0024            CALL VECT (0.,-100.)
          C
          C  CONNECT LINES
          C
0025            CALL APNT (50.,600.,0,-5)
0026            CALL VECT (100.,0.)
0027            CALL VECT (150.,0.,0,-5)
0028            CALL VECT (50.,0.)
0029            CALL VECT (150.,0.,0,-5)
0030            CALL VECT (50.,0.)
0031            CALL VECT (150.,0.,0,-5)
0032            CALL VECT (50.,0.)
0033            CALL VECT (150.,0.,0,-5)
0034            CALL VECT (100.,0.)
          C
          C  LABEL BLOCKS
          C
```

```
0035          CALL APNT (160.,625.,0,-5)
0036          CALL TEXT ('BANDPASS ')
0037          CALL APNT (160.,575.,0,-5)
0038          CALL TEXT ('FILTER ')
0039          CALL APNT (360.,625.,0,-5)
0040          CALL TEXT ('PRODUCT ')
0041          CALL APNT (360.,575.,0,-5)
0042          CALL TEXT ('DETECTOR ')
0043          CALL APNT (560.,625.,0,-5)
0044          CALL TEXT ('INTEGRATE ')
0045          CALL APNT (560.,600.,0,-5)
0046          CALL TEXT ('AND ')
0047          CALL APNT (560.,575.,0,-5)
0048          CALL TEXT ('DUMP ')
0049          CALL APNT (760.,625.,0,-5)
0050          CALL TEXT ('SAMPLE ')
0051          CALL APNT (760.,600.,0,-5)
0052          CALL TEXT ('AND ')
0053          CALL APNT (760.,575.,0,-5)
0054          CALL TEXT ('DECISION ')
0055          CALL APNT (910.,605.,0,-5)
0056          CALL TEXT ('OUTPUT ')
0057          CALL APNT (50.,605.,0,-5)
0058          CALL TEXT ('INPUT ')
      C
      C  LIST ASSUMPTIONS
      C
0059          CALL APNT (50.,450.,0,-5)
0060          CALL TEXT ('ASSUMPTIONS: ')
0061          CALL TEXT (1,'1. BINARY PSK ',1)
0062          CALL TEXT ('2. DETECTION BY A CORRELATION RECEIVER ',1)
0063          CALL TEXT ('3. ADDITIVE WHITE GAUSSIAN NOISE ',1)
0064          CALL TEXT ('4. DECISION THRESHOLD NORMALIZED WITH RESPECT ',
             1       'TO SIGNAL ENERGY ',1)
0065          CALL TEXT ('5. ENERGY CONTRAST RATIO MEASURED AT OUTPUT OF ',
             1       'INTEGRATE AND DUMP ',1)
0066          CALL TEXT ('6. NO INTERSYMBOL INTERFERENCE ')
0067          CALL SAVE ('PSKB2')
0068          CALL INIT (IBUF,1000)
0069          CALL RSTR ('PSKB2')
0070          END
```

```
      DIMENSION IBUF(1000)
      CALL INIT (IBUF,1000)
      CALL APNT (50.0,700.0,0,-5)
C     BEGIN BLOCK 1
      CALL APNT (150.0,550.0,0,-5)
      CALL VECT (150.0,0.0)
      CALL VECT (0.0,100.0)
      CALL VECT (-150.0,0.0)
      CALL VECT (0.0,-100.0)
      PAUSE
C     BEGIN BLOCK 2
      CALL APNT (350.0,550.0,0,-5)
      CALL VECT (150.0,0.0)
      CALL VECT (0.0,100.0)
      CALL VECT (-150.0,0.0)
      CALL VECT (0.0,-100.0)
      PAUSE
C     BEGIN BLOCK 3
      CALL APNT (550.0,550.0,0,-5)
      CALL VECT (150.0,0.0)
      CALL VECT (0.0,100.0)
      CALL VECT (-150.0,0.0)
      CALL VECT (0.0,-100.0)
      PAUSE
C     BEGIN BLOCK 4
      CALL APNT (275.0,425.0,0,-5)
      CALL VECT (100.0,0.0)
      CALL VECT (0.0,100.0)
      CALL VECT (-100.0,0.0)
      CALL VECT (0.0,-100.0)
      PAUSE
C     DRAW CONNECT LINES
      CALL APNT (50.0,600.0,0,-5)
      CALL VECT (100.0,0.0)
      CALL VECT (150.0,0.0,0,-5)
      CALL VECT (50.0,0.0)
      CALL VECT (150.0,0.0,0,-5)
      CALL VECT (50.0,0.0)
      CALL VECT (150.0,0.0,0,-5)
      CALL VECT (100.0,0.0)
      CALL APNT (325.0,600.0,0,-5)
      CALL VECT (0.0,-75.0)
      CALL APNT (375.0,475.0,0,-5)
      CALL VECT (50.0,0.0)
      CALL VECT (0.0,75.0)
      PAUSE
C     LABEL BLOCKS
      CALL APNT (160.0,625.0,0,-5)
      CALL TEXT ('BANDPASS ')
      CALL APNT (160.0,575.0,0,-5)
      CALL TEXT ('FILTER ')
      CALL APNT (360.0,625.0,0,-5)
      CALL TEXT ('COHERENT ')
      CALL APNT (360.0,575.0,0,-5)
```

```
      CALL TEXT ('DETECTOR ')
      CALL APNT (560.0,625.0,0,-5)
      CALL TEXT ('SAMPLE ')
      CALL APNT (560.0,600.0,0,-5)
      CALL TEXT ('AND ')
      CALL APNT (560.0,575.0,0,-5)
      CALL TEXT ('DECISION ')
      CALL APNT (705.0,605.0,0,-5)
      CALL TEXT ('OUTPUT ')
      CALL APNT (50.0,605.0,0,-5)
      CALL TEXT ('INPUT ')
      CALL APNT (285.0,475.0,0,-5)
      CALL TEXT ('DELAY ')
      PAUSE
C     LIST ASSUMPTIONS
      CALL APNT (50.0,400.0,0,-5)
      CALL TEXT ('ASSUMPTIONS: ')
      CALL TEXT (1,'1. BINARY DPSK ',1)
      CALL TEXT ('2. DIFFERENTIAL COHERENT DETECTION ',1)
      CALL TEXT ('3. ADDITIVE WHITE GAUSSIAN NOISE ',1)
      CALL TEXT ('4. NO INTERSYMBOL INTERFERENCE ',1)
      CALL TEXT ('5. DECISION THRESHOLD=0.0 ',1)
      CALL TEXT ('6. MARK PROBABILITY=0.5 ',1)
      CALL TEXT ('7. SIGNAL TO NOISE RATIO MEASURED AT OUTPUT OF  ',
     * 'BANDPASS FILTER ')
      PAUSE
      CALL SAVE ('DPSKB1.DPY')
      PAUSE
      CALL INIT(IBUF,1000)
      CALL RSTR('DPSKB1')
      STOP
      END
```

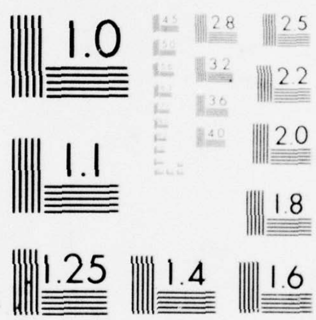
```
0001          DIMENSION IBUF(1000)
0002          CALL INIT (IBUF,1000)
0003          CALL APNT (50.0,700.0,0,-5)
0004          CALL TEXT ('BLOCK DIAGRAM ')
        C     BEGIN BLOCK 1
0005          CALL APNT (150.0,512.0,0,-5)
0006          CALL VECT (150.0,0.0)
0007          CALL VECT (0.0,100.0)
0008          CALL VECT (-150.0,0.0)
0009          CALL VECT (0.0,-100.0)
        C     BEGIN BLOCK 2
0010          CALL APNT (400.0,625.0,0,-5)
0011          CALL VECT (200.0,0.0)
0012          CALL VECT (0.0,25.0)
0013          CALL VECT (-200.0,0.0)
0014          CALL VECT (0.0,-25.0)
        C     BEGIN BLOCK 3
0015          CALL APNT (400.0,575.0,0,-5)
0016          CALL VECT (200.0,0.0)
0017          CALL VECT (0.0,25.0)
0018          CALL VECT (-200.0,0.0)
0019          CALL VECT (0.0,-25.0)
        C     BEGIN BLOCK 4
0020          CALL APNT (400.0,525.0,0,-5)
0021          CALL VECT (200.0,0.0)
0022          CALL VECT (0.0,25.0)
0023          CALL VECT (-200.0,0.0)
0024          CALL VECT (0.0,-25.0)
        C     BEGIN BLOCK 5
0025          CALL APNT (400.0,475.0,0,-5)
0026          CALL VECT (200.0,0.0)
0027          CALL VECT (0.0,25.0)
0028          CALL VECT (-200.0,0.0)
0029          CALL VECT (0.0,-25.0)
        C     BEGIN BLOCK 6
0030          CALL APNT (700.0,512.0,0,-5)
0031          CALL VECT (150.0,0.0)
0032          CALL VECT (0.0,100.0)
0033          CALL VECT (-150.0,0.0)
0034          CALL VECT (0.0,-100.0)
        C     DRAW CONNECT LINES
0035          CALL APNT (50.0,562.0,0,-5)
0036          CALL VECT (100.0,0.0)
0037          CALL VECT (150.0,0.0,0,-5)
0038          CALL VECT (50.0,0.0)
0039          CALL APNT (350.0,487.0,0,-5)
0040          CALL VECT (0.0,150.0)
0041          CALL VECT (50.0,0.0)
0042          CALL APNT (350.0,587.0,0,-5)
0043          CALL VECT (50.0,0.0)
0044          CALL APNT (350.0,537.0,0,-5)
0045          CALL VECT (50.0,0.0)
0046          CALL APNT (350.0,487.0,0,-5)
0047          CALL VECT (50.0,0.0)
```

```
0048        CALL APNT (600.0,487.0,0,-5)
0049        CALL VECT (50.0,0.0)
0050        CALL VECT (0.0,150.0)
0051        CALL VECT (-50.0,0.0)
0052        CALL APNT (600.0,587.0,0,-5)
0053        CALL VECT (50.0,0.0)
0054        CALL APNT (600.0,537.0,0,-5)
0055        CALL VECT (50.0,0.0)
0056        CALL APNT (650.0,562.0,0,-5)
0057        CALL VECT (50.0,0.0)
0058        CALL VECT (150.0,0.0,0,-5)
0059        CALL VECT (100.0,0.0)
      C     LABEL BLOCKS
0060        CALL APNT (50.0,564.0,0,-5)
0061        CALL TEXT ('INPUT ')
0062        CALL APNT (160.0,575.0,0,-5)
0063        CALL TEXT ('BANDPASS ')
0064        CALL APNT (160.0,525.0,0,-5)
0065        CALL TEXT ('FILTER ')
0066        CALL APNT (410.0,627.0,0,-5)
0067        CALL TEXT ('COHERENT DET ')
0068        CALL APNT (410.0,577.0,0,-5)
0069        CALL TEXT ('COHERENT DET ')
0070        CALL APNT (410.0,527.0,0,-5)
0071        CALL TEXT ('COHERENT DET ')
0072        CALL APNT (410.0,477.0,0,-5)
0073        CALL TEXT ('COHERENT DET ')
0074        CALL APNT (710.0,575.0,0,-5)
0075        CALL TEXT ('SAMPLE ')
0076        CALL APNT (710.0,550.0,0,-5)
0077        CALL TEXT ('AND ')
0078        CALL APNT (710.0,525.0,0,-5)
0079        CALL TEXT ('DECISION ')
0080        CALL APNT (860.0,564.0,0,-5)
0081        CALL TEXT ('OUTPUT ')
      C     LIST ASSUMPTIONS
0082        CALL APNT (50.0,400.0,0,-5)
0083        CALL TEXT ('ASSUMPTIONS: ')
0084        CALL TEXT (1,'1. FOUR PHASE PSK (QPSK) ',1)
0085        CALL TEXT ('2. MAXIMUM LIKLIHOOD RECEIVER ',1)
0086        CALL TEXT ('3. ADDITIVE WHITE GAUSSIAN NOISE ',1)
0087        CALL TEXT ('4. NO INTERSYMBOL INTERFERENCE ',1)
0088        CALL TEXT ('5. MARK PROBABILITY=0.5 ',1)
0089        CALL TEXT ('6. SIGNAL TO NOISE RATIO MEASURED AT OUTPUT ',
           *  ' OF BANDPASS FILTER ',1)
0090        CALL SAVE ('QPSKB1')
0091        END
```

AD-A037 833 GEORGIA INST OF TECH ATLANTA ELECTRONICS TECHNOLOGY LAB F/G 9/2
INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY.(U)
FEB 77 R W MOSS, R W RICE, D R SENTZ F30602-75-C-0247
UNCLASSIFIED RADC-TR-77-42 NL

4 OF 4
AD A037833

END
DATE FILMED
4-77



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

```
      DIMENSION IBUF(1000)
      CALL INIT (IBUF,1000)
      CALL APNT (50.0,700.0,0,-5)
      CALL TEXT ('BLOCK DIAGRAM ')
C     BEGIN BLOCK 1
      CALL APNT (200.0,550.0,0,-5)
      CALL VECT (150.0,0.0)
      CALL VECT (0.0,100.0)
      CALL VECT (-150.0,0.0)
      CALL VECT (0.0,-100.0)
C     BEGIN BLOCK 2
      CALL APNT (400.0,550.0,0,-5)
      CALL VECT (150.0,0.0)
      CALL VECT (0.0,100.0)
      CALL VECT (-150.0,0.0)
      CALL VECT (0.0,-100.0)
C     BEGIN BLOCK 3
      CALL APNT (200.0,425.0,0,-5)
      CALL VECT (150.0,0.0)
      CALL VECT (0.0,100.0)
      CALL VECT (-150.0,0.0)
      CALL VECT (0.0,-100.0)
C     BEGIN BLOCK 4
      CALL APNT (400.0,425.0,0,-5)
      CALL VECT (150.0,0.0)
      CALL VECT (0.0,100.0)
      CALL VECT (-150.0,0.0)
      CALL VECT (0.0,-100.0)
C     BEGIN BLOCK 5
      CALL APNT (600.0,425.0,0,-5)
      CALL VECT (150.0,0.0)
      CALL VECT (0.0,225.0)
      CALL VECT (-150.0,0.0)
      CALL VECT (0.0,-225.0)
C     DRAW CONNECT LINES
      CALL APNT (50.0,537.0,0,-5)
      CALL VECT (100.0,0.0)
      CALL APNT (200.0,475.0,0,-5)
      CALL VECT (-50.0,0.0)
      CALL VECT (0.0,125.0)
      CALL VECT (50.0,0.0)
      CALL VECT (150.0,0.0,0,-5)
      CALL VECT (50.0,0.0)
      CALL VECT (150.0,0.0,0,-5)
      CALL VECT (50.0,0.0)
      CALL APNT (350.0,475.0,0,-5)
      CALL VECT (50.0,0.0)
      CALL VECT (150.0,0.0,0,-5)
      CALL VECT (50.0,0.0)
      CALL APNT (750.0,537.0,0,-5)
      CALL VECT (100.0,0.0)
C     LABEL BLOCKS
      CALL APNT (50.0,539.0,0,-5)
      CALL TEXT ('INPUT ')
```

```
0048          CALL APNT (210.0,625.0,0,-5)
0049          CALL TEXT ('BANDPASS ')
0050          CALL APNT (210.0,600.0,0,-5)
0051          CALL TEXT ('FILTER ')
0052          CALL APNT (210.0,575.0,0,-5)
0053          CALL TEXT ('AT F1 ')
0054          CALL APNT (210.0,500.0,0,-5)
0055          CALL TEXT ('BANDPASS ')
0056          CALL APNT (210.0,475.0,0,-5)
0057          CALL TEXT ('FILTER ')
0058          CALL APNT (210.0,450.0,0,-5)
0059          CALL TEXT ('AT F2 ')
0060          CALL APNT (410.0,625.0,0,-5)
0061          CALL TEXT ('ENVELOPE ')
0062          CALL APNT (410.0,575.0,0,-5)
0063          CALL TEXT ('DETECTOR ')
0064          CALL APNT (410.0,500.0,0,-5)
0065          CALL TEXT ('ENVELOPE ')
0066          CALL APNT (410.0,450.0,0,-5)
0067          CALL TEXT ('DETECTOR ')
0068          CALL APNT (610.0,600.0,0,-5)
0069          CALL TEXT ('SAMPLE ')
0070          CALL APNT (610.0,550.0,0,-5)
0071          CALL TEXT ('AND ')
0072          CALL APNT (610.0,500.0,0,-5)
0073          CALL TEXT (' DECISION ')
0074          CALL APNT (760.0,539.0,0,-5)
0075          CALL TEXT ('OUTPUT ')
      C       LIST ASSUMPTIONS
0076          CALL APNT (50.0,400.0,0,-5)
0077          CALL TEXT ('ASSUMPTIONS: ')
0078          CALL TEXT (1,'1. BINARY FSK ',1)
0079          CALL TEXT ('2. NON-COHERENT DETECTION ',1)
0080          CALL TEXT ('3. ADDITIVE WHITE GAUSSIAN NOISE ',1)
0081          CALL TEXT ('4. NO INTERSYMBOL INTERFERENCE ',1)
0082          CALL TEXT ('5. NO FILTER CROSS TALK ',1)
0083          CALL TEXT ('6. SIGNAL TO NOISE RATIO MEASURED AT OUTPUT ',
             *  ' OF BANDPASS FILTER ',1)
0084          CALL SAVE ('FSKB1')
0085          END
```

```
0001          DIMENSION IBUF(1000)
0002          CALL INIT (IBUF,1000)
0003          CALL APNT (50.0,700.0,0,-5)
0004          CALL TEXT ('BLOCK DIAGRAM ')
        C     BEGIN BLOCK 1
0005          CALL APNT (150.0,550.0,0,-5)
0006          CALL VECT (150.0,0.0)
0007          CALL VECT (0.0,100.0)
0008          CALL VECT (-150.0,0.0)
0009          CALL VECT (0.0,-100.0)
        C     BEGIN BLOCK 2
0010          CALL APNT (150.0,425.0,0,-5)
0011          CALL VECT (150.0,0.0)
0012          CALL VECT (0.0,100.0)
0013          CALL VECT (-150.0,0.0)
0014          CALL VECT (0.0,-100.0)
        C     BEGIN BLOCK 3
0015          CALL APNT (350.0,425.0,0,-5)
0016          CALL VECT (150.0,0.0)
0017          CALL VECT (0.0,225.0)
0018          CALL VECT (-150.0,0.0)
0019          CALL VECT (0.0,-225.0)
        C     BEGIN BLOCK 4
0020          CALL APNT (550.0,600.0,0,-5)
0021          CALL VECT (150.0,0.0)
0022          CALL VECT (0.0,50.0)
0023          CALL VECT (-150.0,0.0)
0024          CALL VECT (0.0,-50.0)
        C     BEGIN BLOCK 5
0025          CALL APNT (550.0,500.0,0,-5)
0026          CALL VECT (150.0,0.0)
0027          CALL VECT (0.0,75.0)
0028          CALL VECT (-150.0,0.0)
0029          CALL VECT (0.0,-75.0)
        C     BEGIN BLOCK 6
0030          CALL APNT (550.0,425.0,0,-5)
0031          CALL VECT (150.0,0.0)
0032          CALL VECT (0.0,50.0)
0033          CALL VECT (-150.0,0.0)
0034          CALL VECT (0.0,-50.0)
        C     BEGIN BLOCK 7
0035          CALL APNT (750.0,475.0,0,-5)
0036          CALL VECT (150.0,0.0)
0037          CALL VECT (0.0,125.0)
0038          CALL VECT (-150.0,0.0)
0039          CALL VECT (0.0,-125.0)
        C     DRAW CONNECT LINES
0040          CALL APNT (25.0,537.0,0,-5)
0041          CALL VECT (100.0,0.0)
0042          CALL APNT (150.0,475.0,0,-5)
0043          CALL VECT (-25.0,0.0)
0044          CALL VECT (0.0,125.0)
0045          CALL VECT (25.0,0.0)
0046          CALL APNT (300.0,600.0,0,-5)
```

```
      CALL VECT (50.0,0.0)
      CALL APNT (300.0,475.0,0,-5)
      CALL VECT (50.0,0.0)
      CALL APNT (500.0,625.0,0,-5)
      CALL VECT (50.0,0.0)
      CALL APNT (500.0,450.0,0,-5)
      CALL VECT (50.0,0.0)
      CALL APNT (625.0,475.0,0,-5)
      CALL VECT (0.0,25.0)
      CALL VECT (0.0,75.0,0,-5)
      CALL VECT (0.0,25.0)
      CALL APNT (700.0,625.0,0,-5)
      CALL VECT (125.0,0.0)
      CALL VECT (0.0,-25.0)
      CALL VECT (0.0,-125.0,0,-5)
      CALL VECT (0.0,-25.0)
      CALL VECT (-125.0,0.0)
      CALL APNT (900.0,537.0,0,-5)
      CALL VECT (100.0,0.0)
C     LABEL BLOCKS
      CALL APNT (25.0,539.0,0,-5)
      CALL TEXT ('INPUT ')
      CALL APNT (160.0,625.0,0,-5)
      CALL TEXT ('MATCHED ')
      CALL APNT (160.0,600.0,0,-5)
      CALL TEXT ('FILTER ')
      CALL APNT (160.0,575.0,0,-5)
      CALL TEXT ('AT F1 ')
      CALL APNT (160.0,500.0,0,-5)
      CALL TEXT ('MATCHED ')
      CALL APNT (160.0,475.0,0,-5)
      CALL TEXT ('FILTER ')
      CALL APNT (160.0,450.0,0,-5)
      CALL TEXT ('AT F2 ')
      CALL APNT (360.0,537.0,0,-5)
      CALL TEXT ('SAMPLER ')
      CALL APNT (560.0,610.0,0,-5)
      CALL TEXT ('SUM ')
      CALL APNT (560.0,540.0,0,-5)
      CALL TEXT ('THRESHOLD ')
      CALL APNT (560.0,435.0,0,-5)
      CALL TEXT ('SUM ')
      CALL APNT (760.0,537.0,0,-5)
      CALL TEXT ('DECISION ')
      CALL APNT (900.0,539.0,0,-5)
      CALL TEXT ('OUTPUT ')
C     LIST ASSUMPTIONS
      CALL APNT (50.0,400.0,0,-5)
      CALL TEXT ('ASSUMPTIONS: ')
      CALL TEXT (1,'1. BINARY FSK ',1)
      CALL TEXT ('2. MATCHED FILTER DETECTION ',1)
      CALL TEXT ('3. ADDITIVE WHITE GAUSSIAN NOISE ',1)
      CALL TEXT ('4. NO INTERSYMBOL INTERFERENCE ',1)
      CALL TEXT ('5. NO FILTER CROSSTALK ',1)
```

```
0099      CALL TEXT ('6. ENERGY CONTRAST RATIO MEASURED AT OUTPUT ',
         * ' OF MATCHED FILTER ')
0100      CALL SAVE ('FSKB2')
0101      END
```

```
FORTRAN IV      V01C-03A   WED 04-AUG-76 13:38:15                PAGE 001

          DIMENSION IBUF (1000)
          CALL INIT (IBUF,1000)
          CALL STAT(-1)
          DO 10 I=1,6
          YP=10.0**(I-1)
          DO 20 J=1,9
          Y=FLOAT(J)*YP
          Y=ALOG10(Y)
          Y=100.0*Y+100.0
          CALL APNT (300.0,Y,0,-3)
          IF(J.NE.1) GO TO 99
          CALL VECT (400.0,0.0)
          GO TO 100
       99 CONTINUE
          CALL VECT (10.0,0.0)
      100 CONTINUE
       20 CONTINUE
       10 CONTINUE
          CALL APNT (300.0,700.0,0,-3)
          CALL VECT (400.0,0.0)
          XVAL=300.0
          DO 30 I=1,4
          CALL APNT (XVAL,100.0,0,-3)
          CALL VECT (0.0,600.0)
          DO 40 J=1,4
          XVAL=XVAL+20.0
          CALL APNT (XVAL,100.0,0,-3)
          CALL VECT (0.0,10.0)
       40 CONTINUE
          XVAL=XVAL+20.0
       30 CONTINUE
          CALL APNT (700.0,100.0,0,-3)
          CALL VECT (0.0,600.0)
          CALL APNT (210.0,700.0,0,-3)
          CALL TEXT ('10E+0 ')
          CALL APNT (210.0,600.0,0,-3)
          CALL TEXT ('10E-1 ')
          CALL APNT (210.0,500.0,0,-3)
          CALL TEXT ('10E-2 ')
          CALL APNT (210.0,400.0,0,-3)
          CALL TEXT ('10E-3 ')
          CALL APNT (210.0,300.0,0,-3)
          CALL TEXT ('10E-4 ')
          CALL APNT (210.0,200.0,0,-3)
          CALL TEXT ('10E-5 ')
          CALL APNT (210.0,100.0,0,-3)
          CALL TEXT ('10E-6 ')
          CALL APNT (300.0,75.0,0,-3)
          CALL TEXT ('-10 ')
          CALL APNT (400.0,75.0,0,-3)
          CALL TEXT ('0 ')
          CALL APNT (500.0,75.0,0,-3)
          CALL TEXT ('10 ')
          CALL APNT (600.0,75.0,0,-3)
```

```
      CALL TEXT ('20 ')
      CALL APNT (700.0,75.0,0,-3)
      CALL TEXT ('30 ')
      CALL APNT (320.0,50.0,0,-3)
      CALL TEXT ('SIGNAL TO NOISE RATIO (DB) ')
      CALL APNT (25.0,375.0,0,-3)
      CALL TEXT ('PROBABILITY ')
      CALL APNT (25.0,350.0,0,-3)
      CALL TEXT ('OF ')
      CALL APNT (25.0,325.0,0,-3)
      CALL TEXT ('ERROR ')
      CALL SAVE ('EGRID')
      STOP
      END
```

```
      DIMENSION IBUF (1000)
      CALL INIT (IBUF,1000)
      CALL STAT(-1)
      DO 10 I=1,6
      YP=10.0**(I-1)
      DO 20 J=1,9
      Y=FLOAT(J)*YP
      Y=ALOG10(Y)
      Y=100.0*Y+100.0
      CALL APNT (300.0,Y,0,-3)
      IF(J.NE.1) GO TO 99
      CALL VECT (400.0,0.0)
      GO TO 100
   99 CONTINUE
      CALL VECT (10.0,0.0)
  100 CONTINUE
   20 CONTINUE
   10 CONTINUE
      CALL APNT (300.0,700.0,0,-3)
      CALL VECT (400.0,0.0)
      XVAL=300.0
      DO 30 I=1,4
      CALL APNT (XVAL,100.0,0,-3)
      CALL VECT (0.0,600.0)
      DO 40 J=1,4
      XVAL=XVAL+20.0
      CALL APNT (XVAL,100.0,0,-3)
      CALL VECT (0.0,10.0)
   40 CONTINUE
      XVAL=XVAL+20.0
   30 CONTINUE
      CALL APNT (700.0,100.0,0,-3)
      CALL VECT (0.0,600.0)
      CALL APNT (210.0,700.0,0,-3)
      CALL TEXT ('10E+0 ')
      CALL APNT (210.0,600.0,0,-3)
      CALL TEXT ('10E-1 ')
      CALL APNT (210.0,500.0,0,-3)
      CALL TEXT ('10E-2 ')
      CALL APNT (210.0,400.0,0,-3)
      CALL TEXT ('10E-3 ')
      CALL APNT (210.0,300.0,0,-3)
      CALL TEXT ('10E-4 ')
      CALL APNT (210.0,200.0,0,-3)
      CALL TEXT ('10E-5 ')
      CALL APNT (210.0,100.0,0,-3)
      CALL TEXT ('10E-6 ')
      CALL APNT (300.0,75.0,0,-3)
      CALL TEXT ('-10 ')
      CALL APNT (400.0,75.0,0,-3)
      CALL TEXT ('0 ')
      CALL APNT (500.0,75.0,0,-3)
      CALL TEXT ('10 ')
      CALL APNT (600.0,75.0,0,-3)
```

```
      CALL TEXT ('20 ')
      CALL APNT (700.0,75.0,0,-3)
      CALL TEXT ('30 ')
      CALL APNT (320.0,50.0,0,-3)
      CALL TEXT ('ENERGY CONTRAST RATIO (DB) ')
      CALL APNT (25.0,375.0,0,-3)
      CALL TEXT ('PROBABILITY ')
      CALL APNT (25.0,350.0,0,-3)
      CALL TEXT ('OF ')
      CALL APNT (25.0,325.0,0,-3)
      CALL TEXT ('ERROR ')
      CALL SAVE ('EGRID2')
      STOP
      END
```

```
FORTRAN IV      V01C-03A   WED 04-AUG-76 13:38:24                PAGE 001

      DIMENSION A(8),IBUF(1000)
      CALL INIT(IBUF,1000)
      CALL STAT(-1)
      DO 30 I=1,8
   30 A(I)=100.*ALOG10(FLOAT(I+1))
      CALL APNT(0.,0.,0,-5)
      DO 10 I=1,5
      X=200.+(I-1)*150.
      CALL APNT(X,100.,0,-5)
      CALL LVECT(0.,600.)
   10 CALL RDOT(0.,-600.,0,-5)
      DO 20 I=1,7
      X=100.+(I-1)*100.
      CALL APNT(200.,X,0,-5)
      CALL LVECT(600.,0.)
      CALL RDOT(-600.,0.,0,-5)
      IF(I.EQ.7) GO TO 20
      DO 15 J=1,8
      CALL APNT(200.,X+A(J),0,-5)
   15 CALL VECT(10.,0.)
   20 CONTINUE
      CALL APNT(200.,75.,0,-5)
      CALL TEXT('-10')
      CALL APNT(335.,75.,0,-5)
      CALL TEXT(' 0')
      CALL APNT(500.,75.,0,-5)
      CALL TEXT('10')
      CALL APNT(650.,75.,0,-5)
      CALL TEXT('20')
      CALL APNT(800.,75.,0,-5)
      CALL TEXT('30')
      CALL APNT(300.,50.,0,-5)
      CALL TEXT('SIGNAL TO NOISE RATIO ( DB )')
      CALL APNT(100.,700.,0,-5)
      CALL TEXT('10E-0')
      CALL APNT(100.,600.,0,-5)
      CALL TEXT('10E-1')
      CALL APNT(100.,500.,0,-5)
      CALL TEXT('10E-2')
      CALL APNT(100.,400.,0,-5)
      CALL TEXT('10E-3')
      CALL APNT(100.,300.,0,-5)
      CALL TEXT('10E-4')
      CALL APNT(100.,200.,0,-5)
      CALL TEXT('10E-5')
      CALL APNT(100.,100.,0,-5)
      CALL TEXT('10E-6')
      CALL APNT(0.,350.,0,-5)
      CALL TEXT('PROB. OF',1,'ERROR')
      CALL SAVE('EGRID1')
      STOP
      END
```

```
      DIMENSION A(8),IBUF(1000)
      CALL INIT(IBUF,1000)
      CALL STAT(-1)
      DO 30 I=1,8
   30 A(I)=100.*ALOG10(FLOAT(I+1))
      CALL APNT(0.,0.,0,-5)
      DO 10 I=1,5
      X=200.+(I-1)*150.
      CALL APNT(X,100.,0,-5)
      CALL LVECT(0.,600.)
   10 CALL RDOT(0.,-600.,0,-5)
      DO 20 I=1,7
      X=100.+(I-1)*100.
      CALL APNT(200.,X,0,-5)
      CALL LVECT(600.,0.)
      CALL RDOT(-600.,0.,0,-5)
      IF(I.EQ.7) GO TO 20
      DO 15 J=1,8
      CALL APNT(200.,X+A(J),0,-5)
   15 CALL VECT(10.,0.)
   20 CONTINUE
      CALL APNT(200.,75.,0,-5)
      CALL TEXT('-10')
      CALL APNT(335.,75.,0,-5)
      CALL TEXT(' 0')
      CALL APNT(500.,75.,0,-5)
      CALL TEXT('10')
      CALL APNT(650.,75.,0,-5)
      CALL TEXT('20')
      CALL APNT(800.,75.,0,-5)
      CALL TEXT('30')
      CALL APNT(300.,50.,0,-5)
      CALL TEXT('ENERGY CONTRAST RATIO ( DB )')
      CALL APNT(100.,700.,0,-5)
      CALL TEXT('10E-0')
      CALL APNT(100.,600.,0,-5)
      CALL TEXT('10E-1')
      CALL APNT(100.,500.,0,-5)
      CALL TEXT('10E-2')
      CALL APNT(100.,400.,0,-5)
      CALL TEXT('10E-3')
      CALL APNT(100.,300.,0,-5)
      CALL TEXT('10E-4')
      CALL APNT(100.,200.,0,-5)
      CALL TEXT('10E-5')
      CALL APNT(100.,100.,0,-5)
      CALL TEXT('10E-6')
      CALL APNT(0.,350.,0,-5)
      CALL TEXT('PROB. OF',1,'ERROR')
      CALL SAVE('EGRID3')
      STOP
      END
```

```
      DIMENSION AMP(101),IBUF(1000)
      CALL INIT (IBUF,1000)
      XVAL=100.0
      YVAL=100.0
      DO 20 J=1,16
      CALL APNT (100.0,YVAL,0,-3)
      CALL VECT (500.0,0.0)
      YVAL=YVAL+40.0
   20 CONTINUE
      DO 30 K=1,11
      CALL APNT (XVAL,100.0,0,-3)
      CALL VECT (0.0,600.0)
      XVAL=XVAL+50.0
   30 CONTINUE
      CALL APNT (110.0,50.0,0,-5)
      CALL TEXT ('NORMALIZED FREQUENCY ')
      CALL APNT (10.0,350.0,0,-5)
      CALL TEXT ('MAG(H) ')
      CALL APNT (100.0,75.0,0,-5)
      CALL TEXT ('0 ')
      CALL APNT (200.0,75.0,0,-5)
      CALL TEXT ('1 ')
      CALL APNT (300.0,75.0,0,-5)
      CALL TEXT ('2 ')
      CALL APNT (400.0,75.0,0,-5)
      CALL TEXT ('3 ')
      CALL APNT (500.0,75.0,0,-5)
      CALL TEXT ('4 ')
      CALL APNT (600.0,75.0,0,-5)
      CALL TEXT ('5 ')
      CALL APNT (40.0,100.0,0,-5)
      CALL TEXT ('0.0 ')
      CALL APNT (40.0,300.0,0,-5)
      CALL TEXT ('0.5 ')
      CALL APNT (40.0,500.0,0,-5)
      CALL TEXT ('1.0 ')
      CALL APNT (40.0,700.0,0,-5)
      CALL TEXT ('1.5 ')
      CALL SAVE ('GRID')
      END
```

```
      DIMENSION IBUF(50)
      CALL INIT (IBUF,50)
      CALL SUBP(10)
      CALL APNT (900.0,300.0,1,-5)
      CALL TEXT ('YES ')
      CALL ESUB
      CALL SUBP(11)
      CALL APNT (900.0,200.0,1,-5)
      CALL TEXT ('NO ')
      CALL ESUB
      CALL SAVE ('R3YNO')
      END
```

```
FORTRAN IV      V01C-03A   WED 04-AUG-76 13:38:42            PAGE 001

      DIMENSION IBUF(50)
      CALL INIT (IBUF,50)
      CALL APNT (449.0,385.0,0,-5)
      CALL TEXT ('*WORKING* ')
      CALL SAVE ('R3WAIT')
      END
```

```
      DIMENSION IBUF(500)
      CALL INIT(IBUF,500)
      CALL SUBP(1)
      CALL APNT (100.0,600.0,1,-6)
      CALL TEXT ('ASK (NON-COHERENT) ')
      CALL ESUB
      CALL SUBP(2)
      CALL APNT (100.0,500.0,1,-6)
      CALL TEXT ('ASK (COHERENT) ')
      CALL ESUB
      CALL SUBP(3)
      CALL APNT (100.0,400.0,1,-6)
      CALL TEXT ('ASK (CORRELATION RECEIVER) ')
      CALL ESUB
      CALL SUBP(4)
      CALL APNT (100.0,300.0,1,-6)
      CALL TEXT ('PSK (COHERENT) ')
      CALL ESUB
      CALL SUBP(5)
      CALL APNT (100.0,200.0,1,-6)
      CALL TEXT ('PSK (CORRELATION RECEIVER) ')
      CALL ESUB
      CALL SUBP(6)
      CALL APNT (550.0,600.0,1,-6)
      CALL TEXT ('DPSK ')
      CALL ESUB
      CALL SUBP(7)
      CALL APNT (550.0,500.0,1,-6)
      CALL TEXT ('QPSK ')
      CALL ESUB
      CALL SUBP(8)
      CALL APNT (550.0,400.0,1,-6)
      CALL TEXT ('FSK (NON-COHERENT) ')
      CALL ESUB
      CALL SUBP(9)
      CALL APNT (550.0,300.0,1,-6)
      CALL TEXT ('FSK (MATCHED FILTER) ')
      CALL ESUB
      CALL SAVE ('SLST')
      STOP
      END
```

APPENDIX C

DOCUMENTATION FOR PROGRAM MIDSYS

APPENDIX C

DOCUMENTATION FOR PROGRAM MIDSYS

MIDSYS is a program consisting of a subset of the MIDACS software. MIDSYS requires less main memory than MIDACS, thus allowing room for the GT scroller feature to be used when using a machine with only 16K of main memory. The main program is a copy of subroutine SYSAST described in Appendix A. The subroutines are also copies of subroutines with the same names in Appendix A. The main program is in File SYSPRO.FOR on the DECpack. The save module is named MIDSYS.SAV.

```
      PROGRAM MAIN
      COMMON /CBUF/IBUF(100)
      COMMON/CYC/YC(40)
      COMMON T,J,IB,IQ,IG,ICOM,ICNTRL
      COMMON/CUPB/UPB(40),IBND
      DO 1 I=1,40
    1 UPB(I)=0.
      CALL SYSINT
D     PAUSE 'TYPE RE OR RSU TO CONTINUE'
  100 CALL SLST(M)
      CALL INIT(IBUF,100)
      GO TO (10,20,30,40,50,60,70,80,90) M
   10 CALL AB1
      GO TO 110
   20 CALL AB2
      GO TO 110
   30 CALL AB3
      GO TO 110
   40 CALL PS1
      GO TO 110
   50 CALL PS2
      GO TO 110
   60 CALL DPS
      GO TO 110
   70 CALL QPS
      GO TO 110
   80 CALL FS1
      GO TO 110
   90 CALL FS2
  110 CALL SYSMSS(1)
      CALL R3YNO(I)
      CALL INIT
      IF(I.EQ.11) GO TO 120
      GO TO 100
  120 IF(M.EQ.7) GO TO 155
      IF(M.GT.7) GO TO 150
      IF(M.GE.4) GO TO 140
      CALL PRAB(M)
      GO TO 160
  140 CALL PRPSD(M)
      GO TO 160
  150 CALL PRFS(M)
      GO TO 160
  155 CALL PRQ(M)
  160 CALL CNVT
      IF((M.EQ.3).OR.(M.EQ.5).OR.(M.EQ.9)) GO TO 170
      CALL DSPSNR
      GO TO 180
  170 CALL DSPECR
  180 CALL SYSMSS(2)
      CALL R3YNO(I)
      CALL INIT
      IF(I.EQ.10) GO TO 120
      CALL SYSMSS(3)
```

```
0060        CALL R3YNO(I)
0061        CALL INIT
0062        IF(I.EQ.10) GO TO 100
0064        ICOM=1
0065        ICNTRL=1
0066        STOP
0067        END
```

APPENDIX D

SUBROUTINE CALLING HIERARCHY FOR MIDACS DEMONSTRATION SOFTWARE

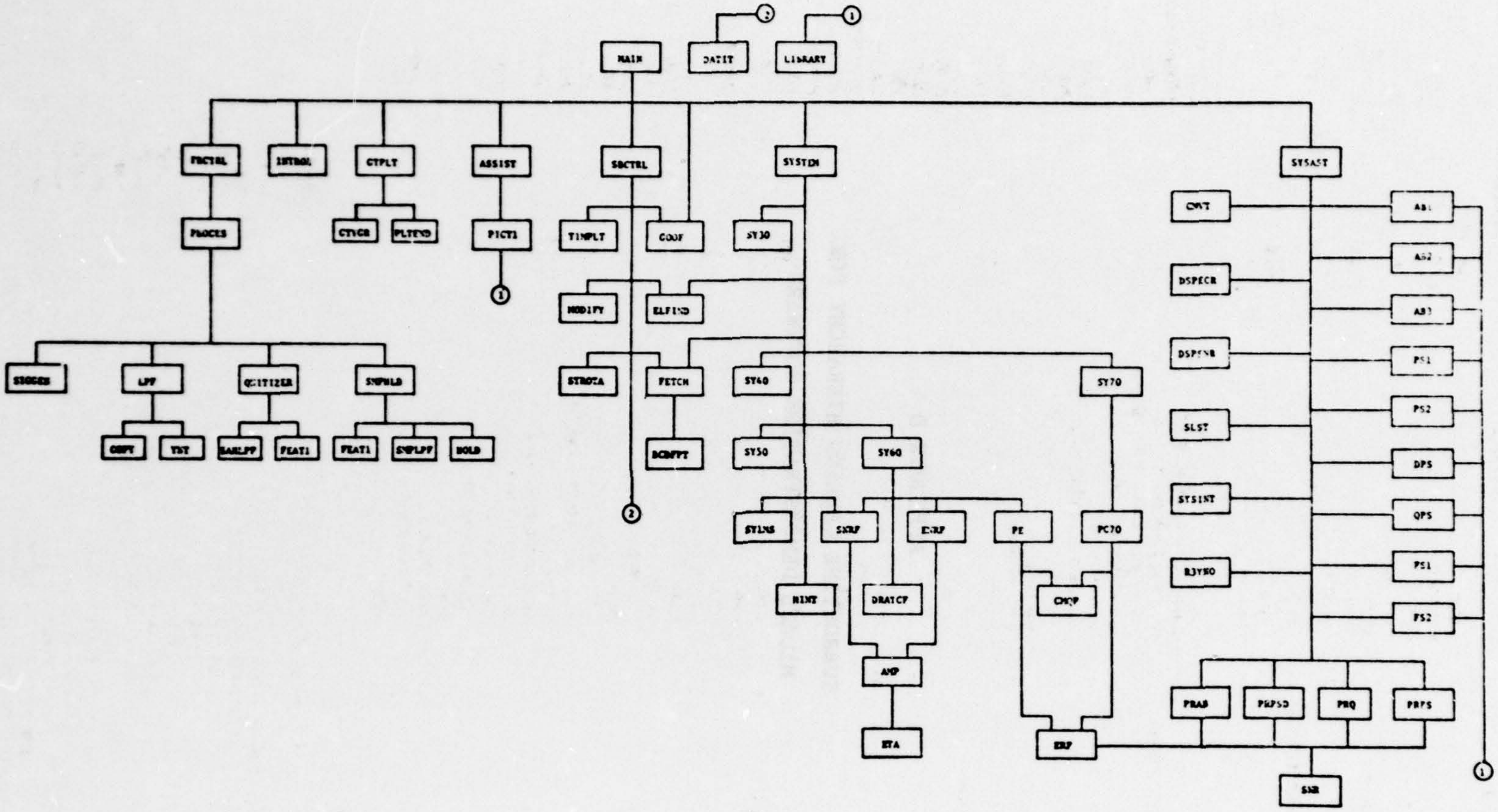


Figure D-1. Subroutine Calling Hierarchy for MIDACS Demonstration Software.

*U.S. GOVERNMENT PRINTING OFFICE: 1977-714-025/138

METRIC SYSTEM

BASE UNITS:

Quantity	Unit	SI Symbol	Formula
length	metre	m	...
mass	kilogram	kg	...
time	second	s	...
electric current	ampere	A	...
thermodynamic temperature	kelvin	K	...
amount of substance	mole	mol	...
luminous intensity	candela	cd	...

SUPPLEMENTARY UNITS:

plane angle	radian	rad	...
solid angle	steradian	sr	...

DERIVED UNITS:

Acceleration	metre per second squared	...	m/s
activity (of a radioactive source)	disintegration per second	...	(disintegration)/s
angular acceleration	radian per second squared	...	rad/s
angular velocity	radian per second	...	rad/s
area	square metre	...	m
density	kilogram per cubic metre	...	kg/m
electric capacitance	farad	F	A·s/V
electrical conductance	siemens	S	A/V
electric field strength	volt per metre	...	V/m
electric inductance	henry	H	V·s/A
electric potential difference	volt	V	W/A
electric resistance	ohm		V/A
electromotive force	volt	V	W/A
energy	joule	J	N·m
entropy	joule per kelvin	...	J/K
force	newton	N	kg·m/s
frequency	hertz	Hz	(cycle)/s
illuminance	lux	lx	lm/m
luminance	candela per square metre	...	cd/m
luminous flux	lumen	lm	cd·sr
magnetic field strength	ampere per metre	...	A/m
magnetic flux	weber	Wb	V·s
magnetic flux density	tesla	T	Wb/m
magnetomotive force	ampere	A	...
power	watt	W	J/s
pressure	pascal	Pa	N/m
quantity of electricity	coulomb	C	A·s
quantity of heat	joule	J	N·m
radiant intensity	watt per steradian	...	W/sr
specific heat	joule per kilogram-kelvin	...	J/kg·K
stress	pascal	Pa	N/m
thermal conductivity	watt per metre-kelvin	...	W/m·K
velocity	metre per second	...	m/s
viscosity, dynamic	pascal-second	...	Pa·s
viscosity, kinematic	square metre per second	...	m/s
voltage	volt	V	W/A
volume	cubic metre	...	m
wavenumber	reciprocal metre	...	(wave)/m
work	joule	J	N·m

SI PREFIXES:

Multiplication Factors	Prefix	SI Symbol
$1\ 000\ 000\ 000\ 000 = 10^{12}$	tera	T
$1\ 000\ 000\ 000 = 10^{9}$	giga	G
$1\ 000\ 000 = 10^{6}$	mega	M
$1\ 000 = 10^{3}$	kilo	k
$100 = 10^{2}$	hecto*	h
$10 = 10^{1}$	deka*	da
$0.1 = 10^{-1}$	deci*	d
$0.01 = 10^{-2}$	centi*	c
$0.001 = 10^{-3}$	milli	m
$0.000\ 001 = 10^{-6}$	micro	μ
$0.000\ 000\ 001 = 10^{-9}$	nano	n
$0.000\ 000\ 000\ 001 = 10^{-12}$	pico	p
$0.000\ 000\ 000\ 000\ 001 = 10^{-15}$	femto	f
$0.000\ 000\ 000\ 000\ 000\ 001 = 10^{-18}$	atto	a

* To be avoided where possible.

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

AMERICAN REVOLUTION BICENTENNIAL 1776-1976

FILM

4